50.005 Computer System Engineering
Quiz OS 3 (30 mins)

Date: 06/03/2020

Name: Poh Shi Hui Student ID: 1002921

Note: This quiz is closed-book and closed-notes, except for one double-sided A4 cheat sheet allowed. You also can't go online or look at anything electronic, including your laptop, smartphone, etc. No calculators are allowed.

Total marks: 15

1. **[1m]** An OS kernel resource manager ensures no deadlocks by checking only that requests are legal *before* granting them, without having to check the system's detailed resource allocation state. This approach of handling deadlocks is called deadlock Avoidance .

2. For each of the following, say whether **it is true or false**:
(a) **[1m]** The Unix file system implements directories as a special type of files. False
(b) **[1m]** In a tree-structured directory, a regular file can have only one path name. False

3. The following snapshot shows the Fd table of two processes at a point in time, as well as a UNIX system wide file table. There are only 3 file descriptors in each of the fd tables of P1 and P2. Note that there may be other processes running in the system as well.

FD	Fd Flags	File Ptr
0	…	3
1	…	5
2	…	1

Fd table P1

FD	Fd Flags	File Ptr
0	…	1
1	…	6
2	…	2

Fd table P2

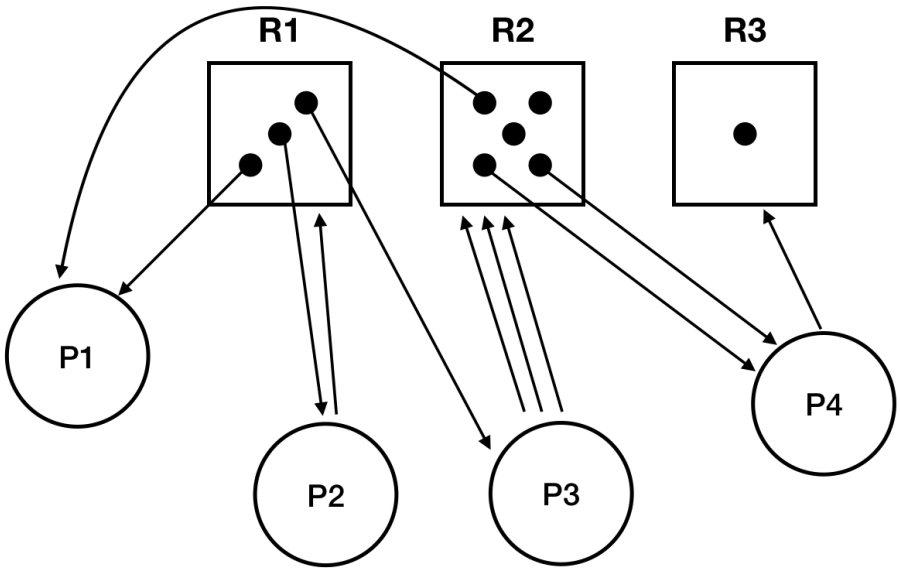
CP	Status Flags	Inode Ptr
33	…	23
25	…	3
57	…	11
0	…	3
14	…	4
15	…	3
16	…	9
	….. (more entries)	

Open File Table

State whether the following statements are **TRUE** or **FALSE** or **MAYBE. Give 1 sentence of explanation to your answer (not more than 15 words).** An answer without explanation will not be granted any marks.

1. **[1m]** P1 and P2 shares **usage** of at least 1 file
   True. Both reference same file ptr 1.
2. **[1m]** P2 is a child process of P1
   False. Fd does not have same content.
3. **[1m]** There exist **different usage instances** of the **same file**
   True. cp 25,0,15 are opening 3 seperately

3. Consider a snapshot of resource allocation graph below and answer the following questions:



(a) **[1m]** Fill in the *Available* matrix based on the state shown in the graph above:

R1	R2	R3
0	2	1

(b) **[2m]** Fill in the *Allocation* matrix based on the state shown in the graph above:

	R1	R2	R3
P1	1	1	0
P2	1	0	0
P3	1	0	0
P4	0	2	0

(c) **[3m]** Given the following *Max* matrix,

	R1	R2	R3
P1	2	2	2
P2	3	0	2
P3	1	3	0
P4	0	2	2

Is the system in the safe state? **Explain your answer.**

Need

	R1	R2	R3
P1	1	1	2
P2	2	0	2
P3	0	3	0
P4	0	0	2

Avail

0	2	1

Not safe.

no need < available.

4. Consider the following Java class:

```
class SampleObject{
    private int x;

    //Constructor
    public SampleObject(int x){
        this.x = x;
    }


    private synchronized void add(int a){
        x += a;
    }

    private synchronized void sub(int a){
        x -= a;
    }

    public synchronized void operation(int a, int b){
        add(a);
        sub(b);
    }

    public synchronized void printx(){
        System.out.println(x);
    }
}
```

An instance of this SampleObject is initialized in the main function, and two threads are created;

```
public static void main(String[] args){
        SampleObject myObject = new SampleObject(10);

        Thread threadOne = new Thread(){
            public void run()
            {
                System.out.println("ThreadOne running");
                myObject.operation(1,1);
            }
        };

        Thread threadTwo = new Thread(){
            public void run()
            {
                System.out.println("ThreadTwo running");
                myObject.operation(1,1);
            }
        };
```

```
        //start each
        threadOne.start();
        threadTwo.start();

        //wait until finish
        try{
            threadOne.join();
            threadTwo.join();
        }
        catch(Exception e){
            e.printStackTrace();
        }

        //print status
        myObject.printx();

        System.out.println("Main thread done.");

    }
```

Each thread performs the `operation` function, which calls `add` and `sub`. Since all functions of `SampleObject` have the synchronized keyword, the threads need to acquire the object lock before accessing them.

[2m] A call on `operation` requires the calling thread to access `add` and `sub` which requires the same object lock. Will this cause any issue? Explain your answer in not more than 15 words. Answer that are > 15 words **will not be graded**.

No. Operation is synchronized and will block the other thread until it is done.

[1m] Write **ALL** possible value(s) of x after the entire program is run. If you think the program will meet a deadlock situation, write `deadlock`.

No deadlock.

X = 10