Subject: Operating System (CS33101)
ASSIGNMENT - 1

NAME: PRASHANT KAUSHIK REG NO.: 2018CA06

Ans 1.

The greatest challenges in designing mobile operating systems include:

- Less storage capacity means the operating system must manage memory carefully.
- The operating system must also manage power consumption carefully.
- Less processing power plus fewer processors mean the operating system must carefully apportion processors to applications.

Ans 2.

An interrupt is a hardware which generate change of flow within the system . An interrupt handler is summoned to deal with the cause of the interrupt; control is then returned to the interrupted context and instruction. A trap is a software-generated interrupt. An interrupt can be used to signal the completion of an I/O to obviate the need for device polling. A trap can be used to call operating system routines or to catch arithmetic errors.

Ans 3.

(a) Synchronous and asynchronous communication : A benefit of synchronous communication is that it allows a rendezvous between the sender and receiver. A disadvantage of a blocking send is that a rendezvous may not be required and the message could be delivered asynchronously. As a result, message-passing systems often provide both forms of synchronization.

(b) Automatic and explicit buffering : Automatic buffering provides a queue with indefinite length, thus ensuring the sender will never have to block while waiting to copy a message. There are no specifications on how automatic buffering will be provided; one scheme may reserve sufficiently large memory where much of the memory is wasted. Explicit buffering specifies how large the buffer is. In this situation, the sender may be blocked while waiting for available space in the queue. However, it is less likely that memory will be wasted with explicit buffering.

(c) Send by copy and send by reference : Send by copy does not allow the receiver to alter the state of the parameter; send by reference does allow it. A benefit of send by reference is that it allows the programmer to write a distributed version of a
centralized application. Java's RMI provides both; however, passing a parameter by reference requires declaring the parameter as a remote object.

(d) Fixed-sized and variable-sized messages : The implications of this are mostly related to buffering issues; with fixed-size messages, a buffer with a specific size can hold a known number of messages. The number of variable-sized messages that can be held by such a buffer is unknown. Consider how Windows 2000 handles this situation with fixed-sized messages (anything < 256 bytes), the messages are copied from the address space of the sender to the address space of the receiving process. Larger messages (i.e. variable-sized messages) use shared memory to pass the message.

Ans 4.

A semaphore can be implemented using the following monitor code:

```
monitor semaphore
{
int value = 0;
condition c;
semaphore increment()
 {
        Value++;
        c.signal();
 }
semaphore decrement()
 {
        while (value == 0)
        c.wait();
        value--;
 }

}
```

Monitor-based Solution to Dining Philosophers :

```
void philosopher(void)
```

```
{
        while(true)
        {
        thinking();
        wait(table_fork(Si));
        wait(table_fork(S(i+1) mod n);
                Eat();                  //critical section
        signal(put_fork(Si));
        signal(put_fork(S(i+1) mod n);
        }
}
```

Ans 5.

(a) Threads count depends upon the priority and requirements of the application. So only thread is enough for this kind of application and this thread is going to handle both input and output operation. It is a concurrency approach. Here, it only makes sense to create as many threads as there are blocking system calls, as the threads will be spent blocking.It doesn't provide any benefits to create an additional threads.Thus, only a signal thread creation makes sense for input and a single thread for Output.

(b) Four threads are created to perform the CPU-intensive portion of the application. It is because, there should be as many threads as there are processing cores.
It would be a waste of processing resources to use fewer threads.
Also any number greater than four would be unable to run.

Ans 6.

Since initial value of semaphore is 2, two processes can enter critical section at a time- this is bad as suppose X and Y be the processes X increments x by 1 and Z decrements x by 2. Now, Z stores back and after this X stores back. So, the final value of x is 1 and not -1 and two Signal operations make the semaphore value 2 again. So, now W and Z can also execute like this and the value of x can be 2 which is the maximum possible in any order of execution of the processes.

(If the semaphore is initialized to 1, processed would execute correctly and we get the final value of x as -2.)

Ans 7.

There are following ways that concurrent processes can follow.

C = B – 1; // C = 1
B = 2*C; // B = 2
D = 2 * B; // D = 4
B = D - 1; // B = 3

C = B – 1; // C = 1
D = 2 * B; // D = 4
B = D - 1; // B = 3
B = 2*C; // B = 2

C = B – 1; // C = 1
D = 2 * B; // D = 4
B = 2*C; // B = 2
B = D - 1; // B = 3

D = 2 * B; // D = 4
C = B – 1; // C = 1
B = 2*C; // B = 2
B = D - 1; // B = 3

D = 2 * B; // D = 4
B = D - 1; // B = 3
C = B – 1; // C = 2
B = 2*C; // B = 4

There are 3 different possible values of B: 2, 3 and 4.

Ans 8.

(a) The time quantum is 1 millisecond: Irrespective of which process is scheduled, the
scheduler incurs a 0.1 millisecond context-switching cost for every context-switch. This
results in a CPU utilization of 1/1.1 * 100 = 91%.

(b) The time quantum is 10 milliseconds: The I/O-bound tasks incur a context switch after using up only 1 millisecond of the time quantum. The time required to cycle through all the processes is therefore 10*1.1 + 10.1 (as each I/O-bound task executes for 1 millisecond and then incur the context switch task, whereas the CPU-bound task executes for 10 milliseconds before incurring a context switch). The CPU utilization is therefore 20/21.1 * 100 = 94%.

Ans 9.

The following rule prevents deadlock: when a philosopher makes a request for the first chopstick, do not grant the request if there is no other philosopher with two chopsticks
and if there is only one chopstick remaining.

Ans 10.

Please note that page fault rate is given one page fault per 10,000 instructions. Since
there are two memory accesses per instruction, so we need double address translation time for average instruction execution time. Also, there are 2 page table accessed if TLB miss occurred. TLB access assumed as 0.

Therefore,
Average Instruction execution time
= Average CPU execution time + Average time for getting data (instruction operands from memory for each instruction)

= Average CPU execution time + Average address translation time for each instruction + Average memory fetch time for each instruction + Average page fault time for each
instruction

= 100 + 2×(0.9×(0)+0.1×(2×150)) + 2×150 + 1 /10000 × 8 × 10 6
= 100 + 60 + 300 + 800
= 1260 ns