Question 1: (3 mark) A program (process-A) is executing on a dual core processor as shown below where there is no other processes running. Mention in which of the 4 states, each processes (A & B) will reside at each time unit **T** mentioned below. Leave the slots empty wherever that state is not applicable for any process.

Process-A
T-0: fork() // Process-B
T-1: fib(20) // Takes T=2
T-2:
T-3:
T-4: wait()
T-5:

Process-B
T-0:
T-1: printf(Hello World)
T-2: scanf(n) // Takes T=1
T-3: fib(n=20) // Takes T=2
T4:
T5:

Time (T)	New	Ready	Running	Waiting
0	B		A	
1			A	B
2			A	
3			A	B
4			B	A
5			B	A

TAs: There is 0.5 mark for each of the three lines of code in Process A & B. Hence, total 0.5x6 marks. Even if any one of the time slot is misfiled for a particular line in the code, then zero marks for that line. It is fine if Process-B is shown in Ready state for T-0 instead of the New state (but only one state).

Question-2: Answer the questions mentioned below for the program shown in Figure-1: a) what is the output of the program? b) Recall, in lecture-6 and lecture-7 it was taught there are 4 life lessons for a process in Unix. b) suggest changes in the program such that all 4 life lessons are followed, and c) What will be the output of the program after those changes (**5 marks**)

```
main( ) {
  printf("Hello\n");
  for(int i=0; i<3; i++) {
    if(fork() == 0) {
      printf("Hi\n");
    }
  }
  printf("World\n");
}
```

Figure-1

a) Hello
Hi
Hi
Hi
Hi
Hi
World
World
World
World
World
World

TA: [0.5 marks] for one "Hello", [0.5 marks] if there are more than 3 "Hi" (any number>3) and [0.5 marks] if there are more than 4 "World" (any number>4). Ordering of prints can be different. It is also fine if some student has mentioned non-deterministic outputs for "Hi" and "World".

b) Parent must call wait (or waitpid) before terminating [+0.5 marks] and total 3 calls should be there [+0.5 marks]. Child process should use exec call [+0.5 marks] to launch another executable to print Hi [+0.5 marks], and Child must call either call exit or return 0 before terminating [+0.5 marks].

c) Hello
Hi
Hi
Hi
World

TA: [+1 marks] is output exactly (even the ordering) as above in part-c.

Question-3: Answer the questions mentioned below for the program shown in Figure-2 that has the "sum" as a variable allocated inside a shared memory (code elided for the same): a) identify the line(s) that is/are the part of critical

```
a)  int sum=0;
b)  main( ) {
c)    for(int i=0; i<100; i++) {
d)      if(fork() == 0) {
e)        sum++;
f)      }
g)    }
h)  }
```

Figure-2

Section, and b) what changes should be done for achieving mutual exclusion
[2 marks]

a) **Line-e is the critical section [+0.5 marks]**
b) **sem_wait before Line-e [+0.75 marks] and**
sem_post after Line-e [0.75 marks]. API names and location must match.