Assignment 4

Que-1:Consider the following code segment:

```
pid_t pid;
pid = fork();
if (pid == 0) { /* child process */
   fork();
   thread _create( . . .);
}fork();
```

a. How many unique processes are created
b. How many unique threads are created?

a) 6 processes

b) 2 threads

Que:2 The program shown in Figure uses the Pthreads API. What would be the output from the program at LINE C and LINE P?

```
#include <pthread.h>
#include <stdio.h>
#include <types.h>
int value = 0;
void *runner(void *param); /* the thread */
int main(int argc, char *argv[])
{
pid_t pid;
pthread t tid;
pthread attr t attr;
     pid = fork();
     if (pid == 0) { /* child process */
         pthread attr init(&attr);
         pthread create(&tid,&attr,runner,NULL);
         pthread join(tid,NULL);
         printf("CHILD: value = %d",value); /* LINE C */
      }
      else if (pid > 0) { /* parent process */
         wait(NULL);
         printf("PARENT: value = %d",value); /* LINE P */
      }
}
void *runner(void *param) { value = 5;
pthread exit(0);
}
```

Ans: Output at LINE C is 5.
Output at LINE P is 0.

Que-3: Consider a multicore system and a multithreaded program written using the many-to many threading model. Let the number of user-level threads in the program be greater than the number of

processing cores in the system. Discuss the performance implications of the following scenarios.

Ans: a. The number of kernel threads allocated to the program is less than the number of processing cores - In this scenario, some of the processors would remain idle since the scheduler maps only kernel threads to processors and not user-level threads to processors.

b. The number of kernel threads allocated to the program is equal to the number of processing cores - In this scenario, it is possible that all of the processors might be utilized simultaneously. However, when a kernel thread blocks inside the kernel the corresponding processor would remain idle.

c. The number of kernel threads allocated to the program is greater than the number of processing cores but less than the number of user-level threads - In this scenario, a blocked kernel thread could be swapped out in favor of another kernel thread that is ready to execute, thereby increasing the utilization of the multiprocessor system.