MID-SEMESTER EXAMINATION, 2015 SOLUTIONS

SUBJECT – OPERATING SYSTEM

1. Consider a typical computer system deadlock scenario and discuss all necessary and sufficient conditions for deadlock.

S: In a deadlock, processes never finish executing and system resources are tied up, preventing other jobs from starting

Example: DEADLOCK with mutex locks

```
/* Create and initialize mutex locks */

    pthread_mutex_t   first_mutex;

    pthread_mutex_t   second_mutex;

    pthread_mutex_init (&first_mutex, NULL);

    pthread_mutex_init (&second_mutex, NULL);
```

Next, two threads are created — thread_one and thread_two and both these threads have accers to mutex locks, thread_one and thread_two run in functions do_work_one() and do_work_two() respectively.

```
/*thread_one runs in this function */

void *do_work_one (void *param)

{ pthread_mutex_lock (&first_mutex);

  pthread_mutex_lock (&second_mutex);

  /* Do some work */

  pthread_mutex_unlock (&second_mutex);

  pthread_mutex_unlock (&first_mutex);

  pthread_exit (0);

}
```

```
/* thread two runs in this fn */

void *do_work_two (void *param)

{ pthread_mutex_lock (&second_mutex);

  pthread_mutex_lock (&first_mutex);

  /* Do some work */

  pthread_mutex_unlock (&first_mutex);

  pthread_mutex_unlock (&second_mutex);

  pthread_exit (0);

}
```

~~KAMLESH~~ Thread_one attempts to acquire locks in order (1) first_mutex (2) second mutex while thread_two does it in reverse order. Deadlock is possible if thread_one acquires first mutex while thread_two **aequits** second_mutex