Write up for Question 2

Steps followed for compiling the kernel and adding system call:

1. Download kernel source code using wget command
2. Extract the tar.gz file
3. Make changes to kernel source code
4. Edit config file so that only required things are compiled
5. Compile the kernel with sudo make -j($number of processors to use)
6. Install modules using : sudo make modules_install install
7. Reboot system

Objective : Modifying the linux kernel source code to support soft real time requirements.

To achieve this a new parameter called *sruntime* in struct *sched_entity* is created which stores the information about the soft runtime of the process (the greater the value the more the priority given to the process). The definition of struct *sched_entity* is in file */include/linux/sched.h*. Initially the value of *sruntime* is set to 0, this is achieved by the change of the init function in */kernel/sched/core.c* to include sruntime also.

Now we change the actual functionality of the CFS scheduler. The CFS scheduler used a red-black tree to store and retrieve information about all the processes. By default the red-black tree picks the process with minimum *vruntime*. We change this so that the red-black tree will first pick the process with the greatest *sruntime* value and if two processes have equal *sruntime* then the one with lower *vruntime* is chosen.

This is achieved by changing the implementation of the function *entity_before()* in file *kernel/sched/fair.c*.

Now the implementation is done such that after each timeslice the *sruntime* value is decremented but the length of the timeslice. This is achieved by changing the implementation of the function *update_curr()* in file *kernel/sched/fair.c* .

Now the next step is defining the System Call called rt_nice to change the value of *sruntime* of a process. In the system call all the information about the process is obtained using its pid in a *task_struct* using *pid_task()* and *find_get_pid()*.
Then the value of *sruntime* of the process is changed to soft runtime given in the input.

Error Handling Done in System Call :

1. Handling the case when soft runtime given in input is negative
   Code :
   ```
   if (p_sruntime < 0) {
        return -EINVAL;
   }
   ```

2. Handling the case when the process is not found
   Code :
   ```
   if (task == NULL) {
        printk("Process not found\n");
        return -ESRCH;
   }
   ```

   Prints “success” if no error is encountered

A program by the name of test.c is used to test if the scheduler functions as desired. The program forks the parent process around 20 times and then for each of then it runs a loop from 0 to 1000000000 and outputs the time it takes to run the loop.

Now this is done again but this time the system call is called before the loop. Now the user can observe the time difference between the process with and without the system call. We can see that the ones with the system call runs faster as compared to the ones without, this is due to the fact that since each of these processes have soft-real time requirements they are executed before any other process and hence require less time to complete.
Sample output :