Q2.
In q2, I have changed the sys.c file and the system call table for x86 of kernel 5.9.1. In the sys.c file, I have defined sh_task_info system call which takes file name and process id as input. For the implementation of this system call, first I have copied the file name into file2 variable using strncpy_from_user so that I can use file2 to call the name of file in rest of my code. I have used pid_task struct to search for the process using it's process id. I have handled the error here and whatever error it gets, it will be stored in the error stack and I can get them using errno.
After that I have used printk to print the value of the 5 field to the terminal. Then i have used filp_open from file struct to create the file or if it is created then to write on it. I have handled error here and returned them. After opening the file, I have used kernel_write to write the all the 5 fields in the file name that we have supplied in the arguments. After that I have used filp_close to close the file.

The user need to give filename and process id as input.

The fields that they are printing is prio, pdeath_signal, normal_prio, exit_signal and static_prio. As these all the fields of integer data field so I have used sprintf to convert the integer into string.

I have handled the error related to filp_open and no_process. Whenever the error occur in system call, then it is returned immediately with non zero value. The error got stored in error stack and can be accessed using errno.