1.)We notice that three signals are sent when the whole code is run.Signals SIGINT,SIGSEGV and SIGSTOP are generated in the main() function by the parent thread and then are further deliver to the child thread tid1,tid2 or tid3.SIGSEGV is sent to thread 1,SIGSTOP is sent to thread 2.Considering the thread tid1 will execute the function func first before thread tid2 or tid3 does.We implemented the lock function in the program thus allowing the tid1 to acquire the lock in func first and sleep and listen for SIGINT while thread tid2 and tid3 remains locked out for 50 seconds(sleep(50); //sleep to catch signals).If the signal SIGINT is not caught within 50 seconds then and thread tid1 is destroyed the lock will remain locked forever.SImilarly if thread tid2 and tid3 is executed first it will acquire the lock and for thread tid2 if the signal SIGINT is caught within 50 seconds it will be ignored since it will run the sig\_func() and for thread tid3 if the signal SIGINT is received, the handler SIG\_IGN will run and ignore the signal for the first time.The second time SIGINT is received, sig\_func2 runs. The remaining signals need only be received for the first time. If SIGSEGV, SIGSTOP or SIGFPE is received, then this thread will run sig\_func. If SIGALRM is received, then this thread will run sig\_func2.

2.)For this section we have adjusted test\_and\_set so that it always return 1.This will causes SIGFPE to be never sent.If SIGFPE is sent,then the thread with tid1 will be stopped and will not receive any signal.Also if SIGFPE is sent we are unable to test lines A-E separately

When we test the whole code together only lines A and B are executed

Line A-pthread\_kill(tid1,SIGSEGV)- send the signals SIGSEGV to the first thread tid1 and is handled by signal handler sig\_func.This signal is generated by parent thread and is delivered to the thread tid1. Thread tid1 handles the signal by ignoring it.

Line B-pthread\_kill(tid2,SIGSTOP)-sends the signals SIGSTOP to the thread tid2.In our code this signal is generated and sent by the parent thread and delivered to thread tid2. At the time of creation of thread tid2,the main process has registered signal SIGSTOP under the signal handler sig\_func().However the signal SIGSTOP is sent to every thread in the process and the whole process is stopped and therefore it cannot be caught by any other handler other than the default handler. After the default signal has handled the received signal it will terminated the entire process pid.Therefore we can say that the process pid and threads tid1,tid2,tid3 will be terminated.One more thing to be noted is that the default signal handler for SIGSTOP cannot be overwritten and therefore default signal handler is always kept for SIGSTOP and this signal will not be handled by sig\_func.

Line C(1)- alarm(3);

Line C(2)- while( !alrmflag ) pause();

Line C(1) sets up for a signal SIGALRM. An alarm goes off in 3 seconds causing the signal SIGALRM to be raised and it is handled by signal handler sig\_func2.Thread tid3 is

listening for this alarm, so it will execute sig\_func2 therefore setting the global variable alrmflag to 1 which had an initial value set to 0

Line C(2) Setting up alrmflag to 1 and sending signal SIGALRM is important otherwise pause() function would be executed. Pause() function works by suspending the execution and blocks until a signal arrives which executes a handler function.Since alrmflag is set to 1 after the alarm in Line C(1) goes off, the program breaks out of the while loop and moves on.

LIne D:pthread\_kill(tid1,SIGINT)

Signal SIGINT is generated by user input from the keyboard like (Ctrl+C) and is generated by process PID and sent to thread tid1.Since this is the first SIGINT signal that tid1 receives, the signal is ignored by tid1 because of the signal handler SIG\_IGN.Although this part is caught by the signal handler but it does not do anything.

Line E:pthread\_kill(tid3,SIGINT)

Another Signal SIGINT is generated by user input from the keyboard like (Ctrl+C) and is generated by process PID and sent to thread tid3.We know that all these threads share the same memory space so therefore tid3 is no longer listening for SIGINT when the signal is called for the first time.Therefore, when tid3 receives a SIGINT, the thread runs sig\_func2.

3.) Considering that the first signal SIGINT is sent five times we notice that the first time SIGINT is raised it will be caught by the first call to the signal() function.Signal (SIGINT,SIG\_IGN) will cause it to ignore the first time.The second time when SIGINT is raised the signal will be handled by the thread tid3 causing the function sig\_func2() to be executed.Furthermore, executing sig\_func2 will cause the alrmflag to be set to 1.Since alrmflag is set to 1 the process will break out of the while loop and continue executing the rest of the code.Also, we had a signal which was caught and the signal handler function returns this will cause the the process to get out of the while loop and stop pausing in the main function.The rest of the signals SIGSEGV, SIGQUIT, SIGSTOP,SIGKILL would be executed in the similar way but they would get past the pause()function,unpausing the program only to return to the while loop and pause again.