1. **Write a C program which blocks SIGOUIT signal for 5 seconds. After 5 second process checks any occurrence of quit signal during this period, if so, it unblock the signal. Now another occurrence of quit signal terminates the program. (Use sigprocmask() and sigpending() )**

#include<stdio.h>

#include<signal.h>

#include<stdlib.h>

static void sig\_quit(int);

int main(void)

{

sigset\_t

newmask, oldmask, pendmask;

if (signal(SIGQUIT, sig\_quit) == SIG\_ERR)

printf("can't catch SIGQUIT");

/\*

\* \* Block SIGQUIT and save current signal mask.

\* \*/

sigemptyset(&newmask);

sigaddset(&newmask, SIGQUIT);

if (sigprocmask(SIG\_BLOCK, &newmask, &oldmask) < 0)

printf("SIG\_BLOCK error");

sleep(5);

/\* SIGQUIT here will remain pending \*/

if (sigpending(&pendmask) < 0)

printf("sigpending error");

if (sigismember(&pendmask, SIGQUIT))

printf("\nSIGQUIT pending\n");

/\*

\* \* Reset signal mask which unblocks SIGQUIT.

\* \*/

if (sigprocmask(SIG\_SETMASK, &oldmask, NULL) < 0)

printf("SIG\_SETMASK error");

printf("SIGQUIT unblocked\n");

sleep(5);

exit(0);

/\* SIGQUIT here will terminate with core file \*/

}

static void

sig\_quit(int signo)

{

printf("caught SIGQUIT\n");

if (signal(SIGQUIT, SIG\_DFL) == SIG\_ERR)

printf("can't reset SIGQUIT");

}