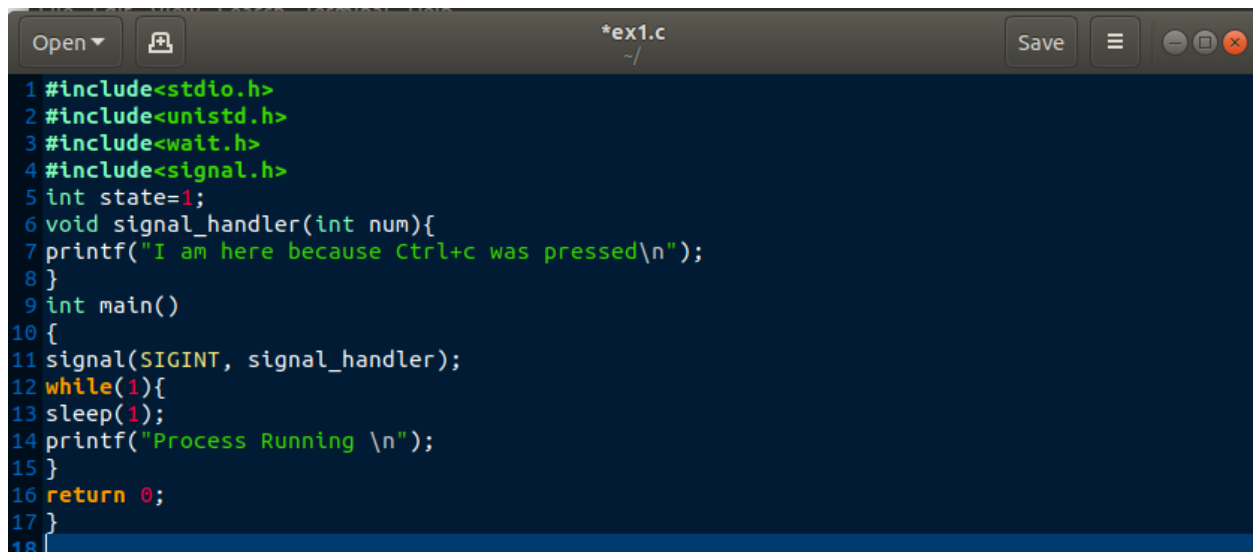


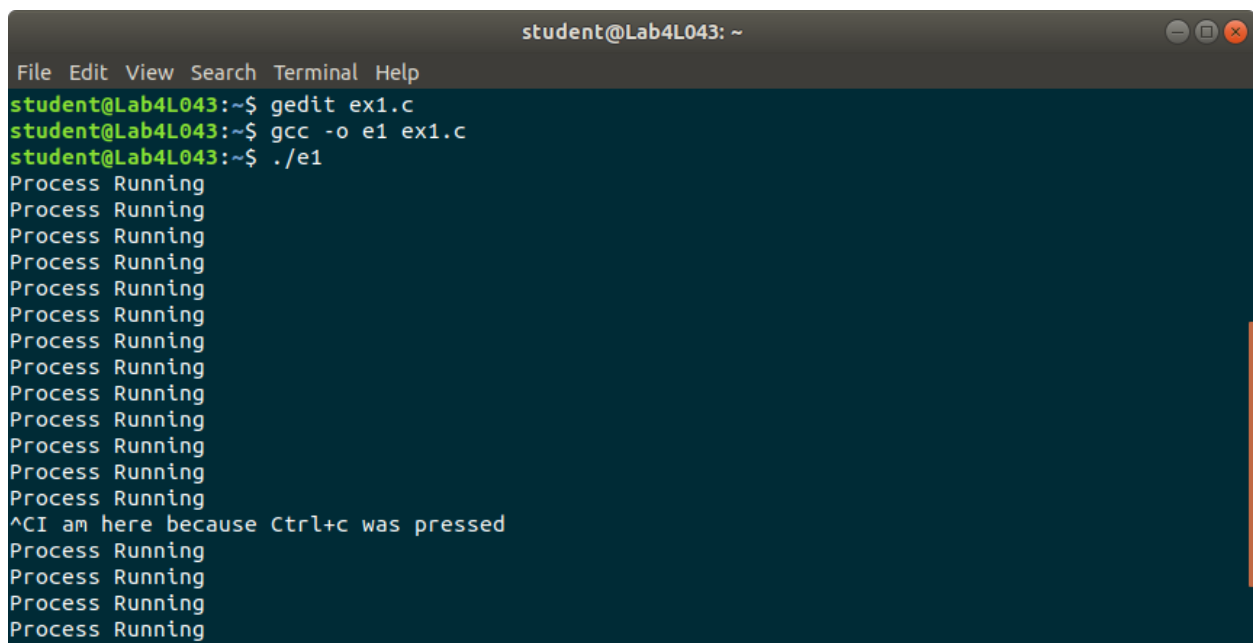
AYAZ HASAN  
20K-1044  
BS-SE (5A)

## LAB9

Q1

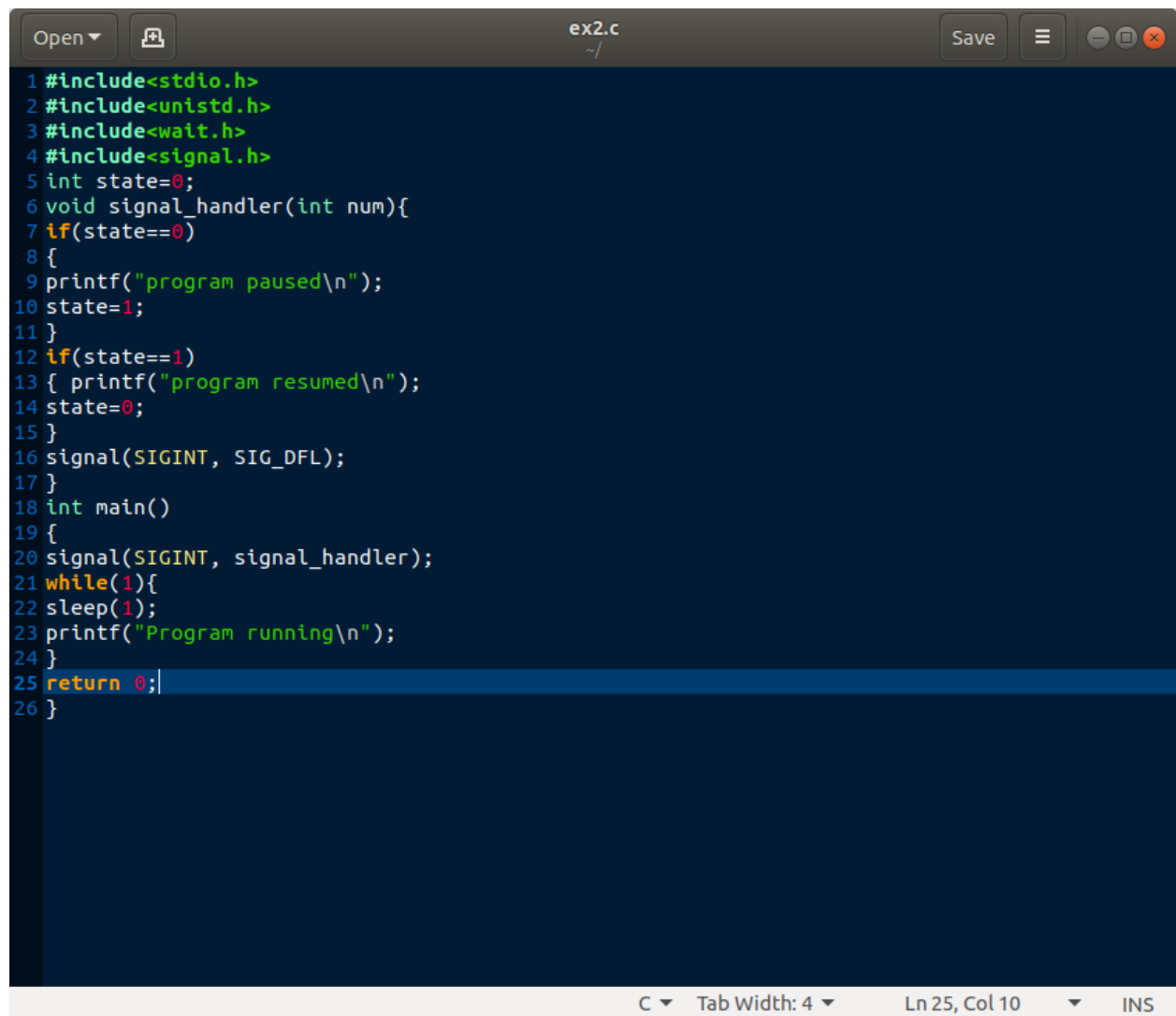


```
1 #include<stdio.h>
2 #include<unistd.h>
3 #include<wait.h>
4 #include<signal.h>
5 int state=1;
6 void signal_handler(int num){
7     printf("I am here because Ctrl+c was pressed\n");
8 }
9 int main()
10 {
11     signal(SIGINT, signal_handler);
12     while(1){
13         sleep(1);
14         printf("Process Running \n");
15     }
16     return 0;
17 }
18 }
```



```
student@Lab4L043: ~
File Edit View Search Terminal Help
student@Lab4L043:~$ gedit ex1.c
student@Lab4L043:~$ gcc -o e1 ex1.c
student@Lab4L043:~$ ./e1
Process Running
Process Running
Process Running
Process Running
Process Running
Process Running
Process Running
Process Running
Process Running
Process Running
Process Running
Process Running
Process Running
Process Running
Process Running
Process Running
^CI am here because Ctrl+c was pressed
Process Running
Process Running
Process Running
Process Running
```

Q2



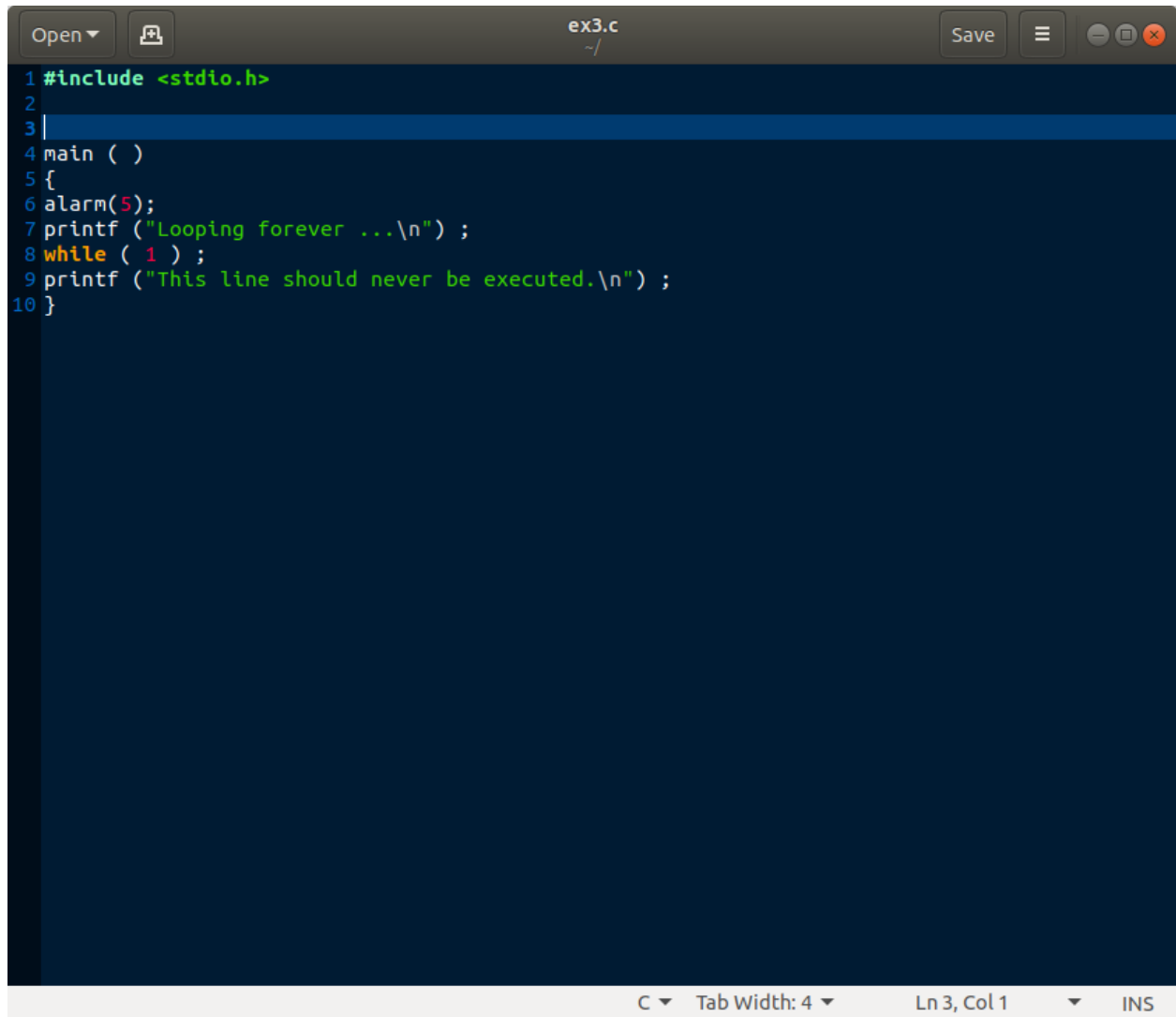
```
1 #include<stdio.h>
2 #include<unistd.h>
3 #include<wait.h>
4 #include<signal.h>
5 int state=0;
6 void signal_handler(int num){
7     if(state==0)
8     {
9         printf("program paused\n");
10        state=1;
11    }
12    if(state==1)
13    { printf("program resumed\n");
14      state=0;
15    }
16    signal(SIGINT, SIG_DFL);
17 }
18 int main()
19 {
20     signal(SIGINT, signal_handler);
21     while(1){
22         sleep(1);
23         printf("Program running\n");
24     }
25     return 0;
26 }
```

Open ▾ ex2.c Save ▮ - □ ×

C ▾ Tab Width: 4 ▾ Ln 25, Col 10 ▾ INS

```
student@Lab4L043:~$ gcc -o e2 ex2.c
student@Lab4L043:~$ ./e2
Program running
Program running
Program running
Program running
^Cprogram paused
program resumed
Program running
Program running
Program running
Program running
Program running
^C
```

Q3



```
1 #include <stdio.h>
2
3
4 main ( )
5 {
6     alarm(5);
7     printf ("Looping forever ...\n") ;
8     while ( 1 ) ;
9     printf ("This line should never be executed.\n") ;
10 }
```

C Tab Width: 4 Ln 3, Col 1 INS

```
student@Lab4L043:~$ gedit ex3.c
student@Lab4L043:~$ gcc -o e3 ex3.c
ex3.c:4:1: warning: return type defaults to 'int' [-Wimplicit-int]
main ( )
^~~~~
ex3.c: In function 'main':
ex3.c:6:1: warning: implicit declaration of function 'alarm' [-Wimplicit-function-declaration]
alarm(5);
^~~~~
student@Lab4L043:~$ ./e3
Looping forever ...
Alarm clock
student@Lab4L043:~$ gedit ex3.c
```

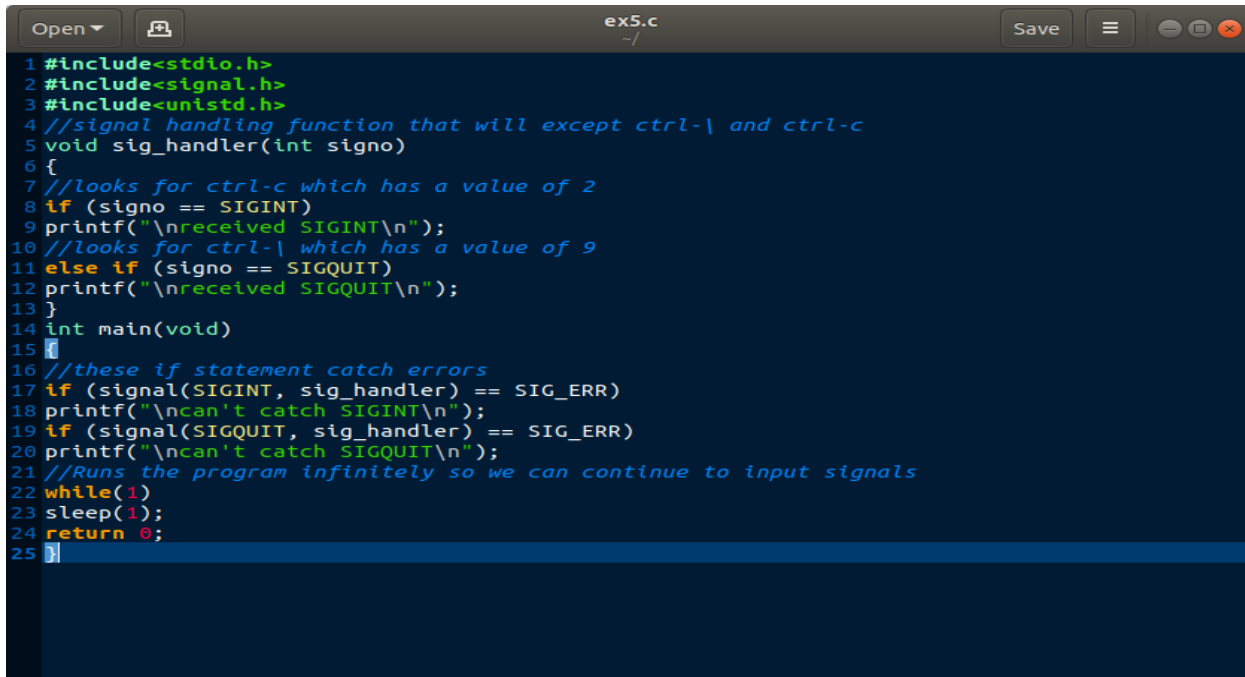
Q4



```
1 #include <stdio.h> #include <signal.h>
2 #include <wait.h>
3 int alarmFlag = 0 ; void alarmHandler ( ) ;
4 main ( ) {
5     signal(SIGALRM, alarmHandler) ; /*Install signal Handler*/ alarm (5) ;
6     printf ("Looping ...\n") ; while (!alarmFlag) {
7         pause ( ) ; /* wait for a signal */
8         printf ("Loop ends due to alarm signal\n");
9     }
10 void alarmHandler ( ) {
11     printf ("An ALARM clock signal was received\n"); alarmFlag = 1;
12 }
```

```
student@Lab4L043:~$ gedit ex4.c
student@Lab4L043:~$ gcc -o e4 ex4.c
ex4.c:1:20: warning: extra tokens at end of #include directive
#include <stdio.h> #include <signal.h>
                    ^
ex4.c:4:1: warning: return type defaults to 'int' [-Wimplicit-int]
main ( ) {
    ^~~~~
ex4.c: In function 'main':
ex4.c:5:60: warning: implicit declaration of function 'alarm' [-Wimplicit-function-declaration]
    signal(SIGALRM, alarmHandler) ; /*Install signal Handler*/ alarm (5) ;
                                                                ^~~~~~
ex4.c:7:1: warning: implicit declaration of function 'pause'; did you mean 'raise'? [-Wimplicit-function-declaration]
    pause ( ) ; /* wait for a signal */
    ^~~~~~
    raise
student@Lab4L043:~$ ./e4
Looping ...
An ALARM clock signal was received
Loop ends due to alarm signal
student@Lab4L043:~$ gedit ex4.c
```

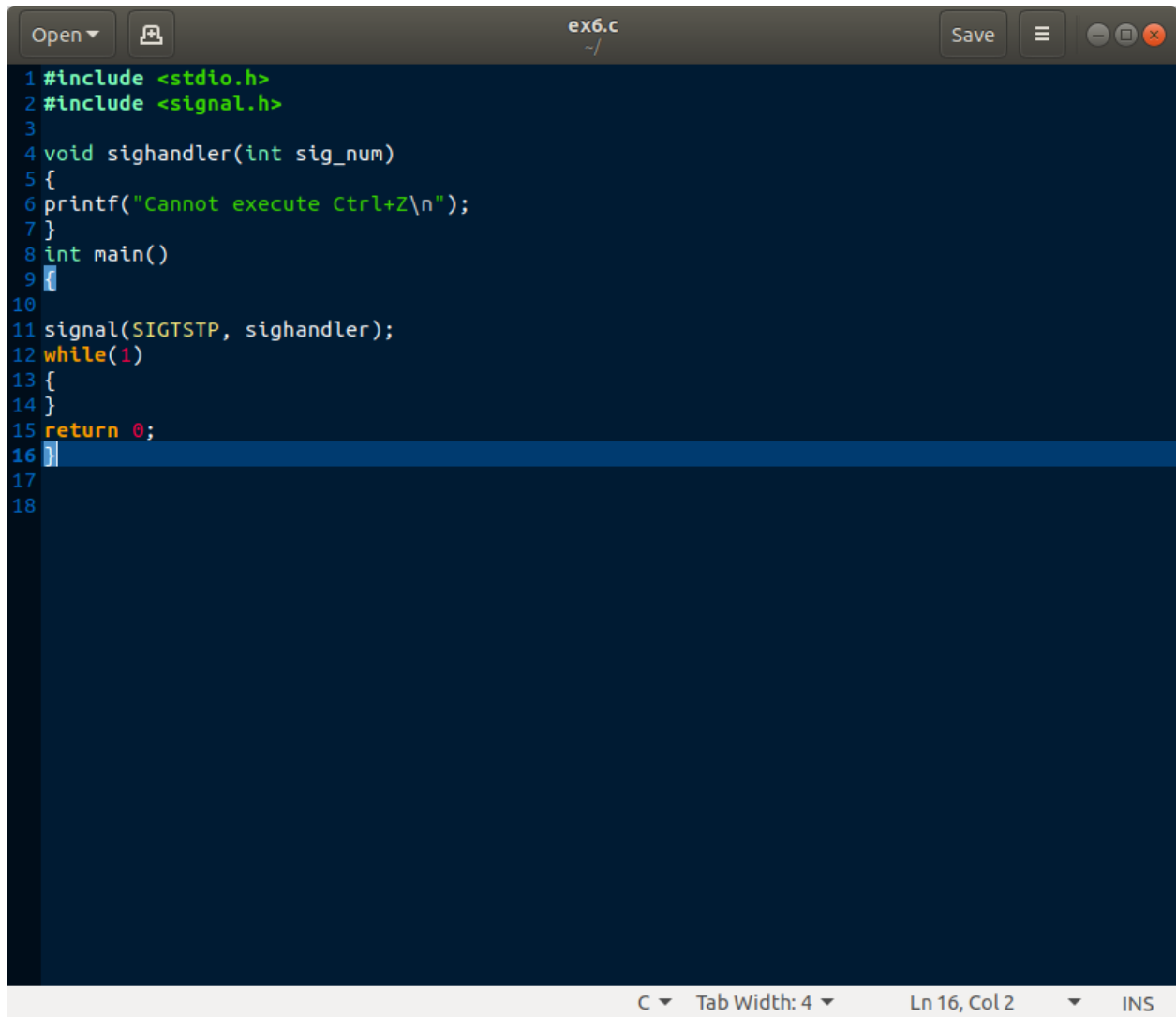
Q5



```
1 #include<stdio.h>
2 #include<signal.h>
3 #include<unistd.h>
4 //signal handling function that will except ctrl-\ and ctrl-c
5 void sig_handler(int signo)
6 {
7     //looks for ctrl-c which has a value of 2
8     if (signo == SIGINT)
9         printf("\nreceived SIGINT\n");
10    //looks for ctrl-\ which has a value of 9
11    else if (signo == SIGQUIT)
12        printf("\nreceived SIGQUIT\n");
13    }
14 int main(void)
15 {
16    //these if statement catch errors
17    if (signal(SIGINT, sig_handler) == SIG_ERR)
18        printf("\ncan't catch SIGINT\n");
19    if (signal(SIGQUIT, sig_handler) == SIG_ERR)
20        printf("\ncan't catch SIGQUIT\n");
21    //Runs the program infinitely so we can continue to input signals
22    while(1)
23        sleep(1);
24    return 0;
25 }
```

```
student@Lab4L043:~$ gedit ex5.c
student@Lab4L043:~$ gcc -o e5 ex5.c
student@Lab4L043:~$ ./e5
^C
received SIGINT
^C
received SIGINT
^Z
[2]+  Stopped                  ./e5
```

Q6



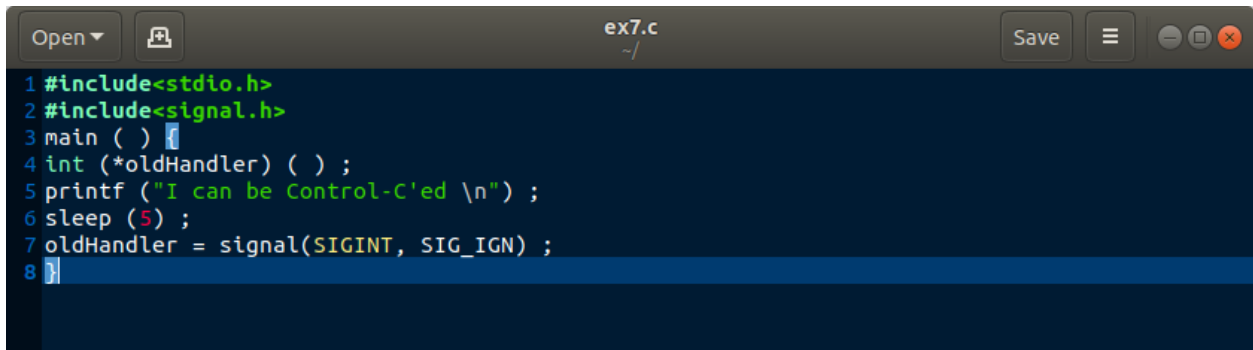
```
1 #include <stdio.h>
2 #include <signal.h>
3
4 void sighandler(int sig_num)
5 {
6     printf("Cannot execute Ctrl+Z\n");
7 }
8 int main()
9 {
10
11     signal(SIGTSTP, sighandler);
12     while(1)
13     {
14     }
15     return 0;
16 }
```

C Tab Width: 4 Ln 16, Col 2 INS

```
Col student@Lab4L043:~$ gedit ex6.c
student@Lab4L043:~$ gcc -o e6 ex6.c
student@Lab4L043:~$ ./e6
^ZCannot execute Ctrl+Z
^ZCannot execute Ctrl+Z
^C
student@Lab4L043:~$ gedit ex6.c
```

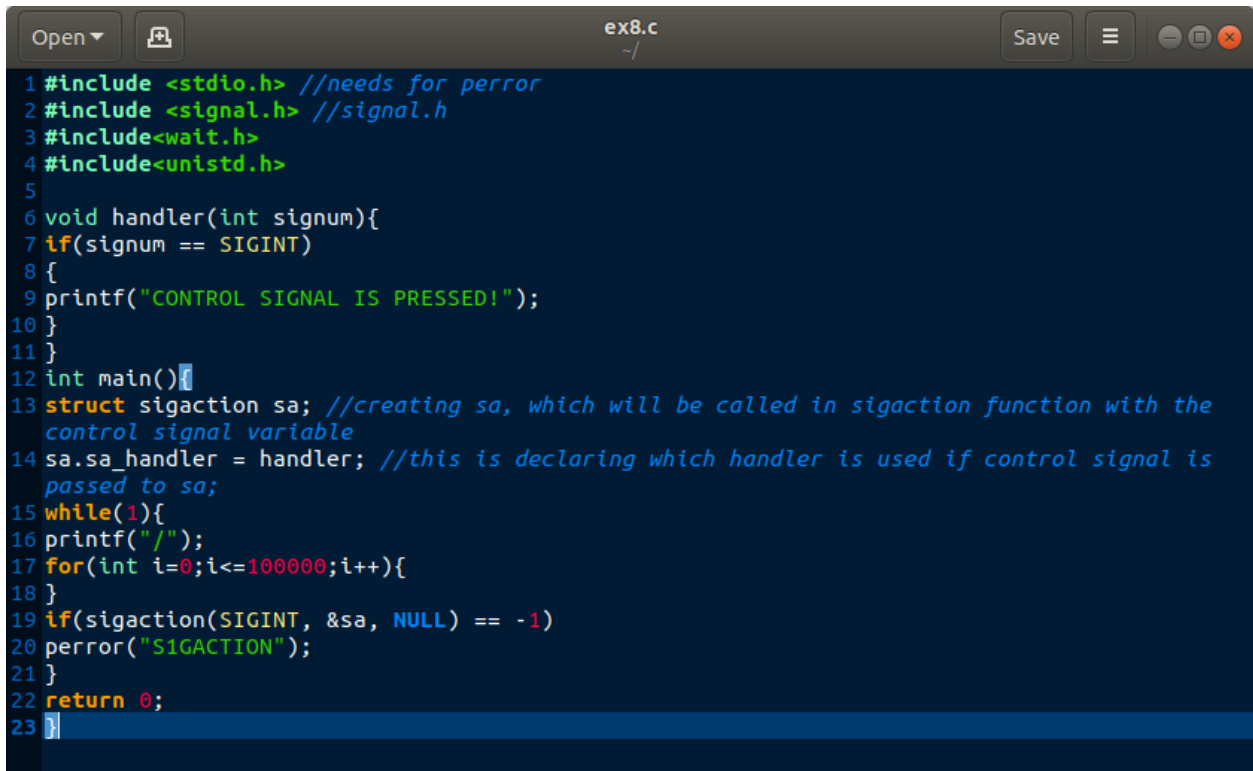
Q7

```
student@Lab4L043:~$ ./e7
I can be Control-C'ed
```



```
1 #include<stdio.h>
2 #include<signal.h>
3 main ( ) {
4 int (*oldHandler) ( ) ;
5 printf ("I can be Control-C'ed \n") ;
6 sleep (5) ;
7 oldHandler = signal(SIGINT, SIG_IGN) ;
8 }
```

Q8



```
1 #include <stdio.h> //needs for perror
2 #include <signal.h> //signal.h
3 #include<wait.h>
4 #include<unistd.h>
5
6 void handler(int signum){
7 if(signum == SIGINT)
8 {
9 printf("CONTROL SIGNAL IS PRESSED!");
10 }
11 }
12 int main(){
13 struct sigaction sa; //creating sa, which will be called in sigaction function with the
    control signal variable
14 sa.sa_handler = handler; //this is declaring which handler is used if control signal is
    passed to sa;
15 while(1){
16 printf("/");
17 for(int i=0;i<=100000;i++){
18 }
19 if(sigaction(SIGINT, &sa, NULL) == -1)
20 perror("SIGACTION");
21 }
22 return 0;
23 }
```

```
student@Lab4L043:~$ ./e8
```

```

^C
CONTROL SIGNAL IS PRESSED!

^C
CONTROL SIGNAL IS PRESSED!

```



## TASK1

```
Open  *task1.c  Save  -  □  ×

1 #include<stdio.h>
2 #include<signal.h>
3 #include<unistd.h>
4
5 void handler(int signo)
6 {
7     if (signo == SIGUSR1)
8         printf("received SIGUSR1\n");
9     else if (signo == SIGKILL)
10        printf("received SIGKILL\n");
11    else if (signo == SIGSTOP)
12        printf("received SIGSTOP\n");
13 }
14
15 int main(void)
16 {
17     struct sigaction sa;
18     sa.sa_handler=handler;
19     if(sigaction(SIGUSR1,&sa,NULL)==-1)
20         printf("\ncan't catch SIGUSR1\n");
21     if(sigaction(SIGKILL,&sa,NULL)==-1)
22         printf("\ncan't catch SIGKILL\n");
23     if(sigaction(SIGSTOP,&sa,NULL)==-1)
24         printf("\ncan't catch SIGSTOP\n");
25
26     while(1)
27         sleep(1);
28     return 0;
29 }
30
31
32

C  Tab Width: 8  Ln 2, Col 19  INS
```

```
student@student-OptiPlex-7090:~$ gedit task1.c
student@student-OptiPlex-7090:~$ gcc -o ob1 task1.c
student@student-OptiPlex-7090:~$ ./ob1

can't catch SIGKILL

can't catch SIGSTOP
^Z
[3]+  Stopped                  ./ob1
student@student-OptiPlex-7090:~$ gedit task1.c
```

## TASK2



```
1 #include<stdio.h>
2 #include<signal.h>
3 #include<unistd.h>
4
5 void handler(int signo)
6 {
7 |
8 if (signo == SIGKILL)
9 printf("received SIGKILL\n");
10 else if (signo == SIGSTOP)
11 printf("received SIGSTOP\n");
12 }
13
14 int main(void)
15 {
16
17     struct sigaction sa;
18
19     sa.sa_handler=handler;
20     if(sigaction(SIGKILL,&sa,NULL)==-1)
21         printf("\ncan't ignore SIGKILL\n");
22     if(sigaction(SIGSTOP,&sa,NULL)==-1)
23         printf("\ncan't ignore SIGSTOP\n");
24
25     while(1)
26         sleep(1);
27     return 0;
28 }
29
```

C ▾ Tab Width: 8 ▾ Ln 7, Col 1 ▾ INS

```
[1] 17082
student@student-OptiPlex-7090:~$ gedit task2.c
student@student-OptiPlex-7090:~$ gcc -o ob2 task2.c
student@student-OptiPlex-7090:~$ ./ob2

can't ignore SIGKILL

can't ignore SIGSTOP
█
```

### TASK3

```
#include <signal.h>
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>

typedef struct data {
    int *arr;
    int thread_num;
} data;

int arrsize= 10;
int sum = 0;
sem_t sem;

void *halfSum(void *p){
    signal(SIGALRM, SIG_IGN);
    data *ptr =(data *)p;
    int n = ptr->thread_num;
    sem_wait(&sem);
    if(n == 0){
        for(int i =0; i < arrsize/2; i++)
        {
            sum +=ptr->arr[i];
        }
    }
    else{
        for(int i =arrsize; i < arrsize/2; i++){
            sum+= ptr->arr[i];
        }
    }
    alarm(5);
    sleep(5);
    sem_post(&sem);
    pthread_exit(NULL);
}

void sig_handler(){
    printf("handler\n");
}
```

```

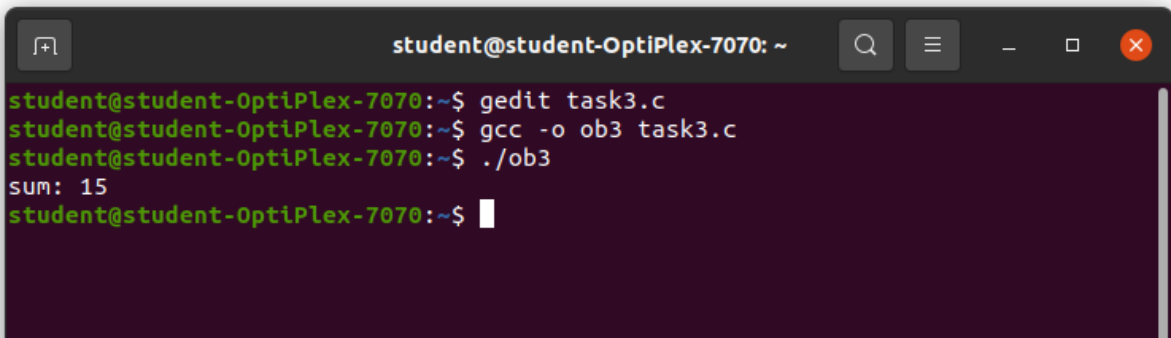
}
int main(){
    int*int_arr=(int *)calloc(arrsize,sizeof(int));
    sem_init(&sem,0,1);
    for(int i = 0; i<arrsize; i++){
        int_arr[i] = i+1;
    }
    data thread_data[2];
    thread_data[0].thread_num = 0;
    thread_data[0].arr=int_arr;
    thread_data[1].thread_num = 1;
    thread_data[1].arr=int_arr;

    pthread_t tid [2];
    pthread_create(&tid[0], NULL,halfSum, &thread_data[0]);
    pthread_create(&tid[1], NULL,halfSum, &thread_data[1]);

    pthread_join(tid[0],NULL);
    pthread_join(tid[1],NULL);
    sem_close(&sem);

    printf("sum: %d\n", sum);
}

```



```

student@student-OptiPlex-7070: ~
student@student-OptiPlex-7070:~$ gedit task3.c
student@student-OptiPlex-7070:~$ gcc -o ob3 task3.c
student@student-OptiPlex-7070:~$ ./ob3
sum: 15
student@student-OptiPlex-7070:~$

```