1. Write a thread program to do the following. Let thread performs summation of cat1 and cat2 mark of a student and return the sum to the main function. Display the name and sum of cat1 and cat2 mark in main().

```
pv@pv-Vostro-5402: /media/pv/Personal/Code/C C++/os_lab_observations
pv@pv-Vostro-5402:/media/pv/Personal/Code/C C++/os_lab_observations$ gedit thread_return_5095.c
^C
pv@pv-Vostro-5402:/media/pv/Personal/Code/C C++/os_lab_observations$ gcc thread_return_5095.c
pv@pv-Vostro-5402:/media/pv/Personal/Code/C C++/os_lab_observations$ ./a.out
Pavana Vamsi
48
46
Name:Pavana Vamsi
Total(before join):0
Total(after join):94
pv@pv-Vostro-5402:/media/pv/Personal/Code/C C++/os_lab_observations$ cat thread_return_5095.c
#include<stdio.h>
#include<pthread.h>
struct student
{
char  name[20];
int cat1;
int cat2;
int total;
}s1;
void * totalcal(void * st)
{
struct student * s2=(struct student *)st;
s2->total=s2->cat1+s2->cat2;
pthread_exit(NULL);
}
int main()
{
scanf("%[^\n]s",s1.name);
scanf("%d",&s1.cat1);
scanf("%d",&s1.cat2);
pthread_t t1;
pthread_create(&t1,NULL,(void*)totalcal,(void*)&s1);
printf("Name:%s\n",s1.name);
printf("Total(before join):%d\n",s1.total);
pthread_join(t1,NULL);
printf("Total(after join):%d\n",s1.total);
return 0;
}

pv@pv-Vostro-5402:/media/pv/Personal/Code/C C++/os_lab_observations$
```

2. Implement the question 1 for array of 5 students.

```
pv@pv-Vostro-5402: /media/pv/Personal/Code/C C++/os_lab_observations
pv@pv-Vostro-5402:/media/pv/Personal/Code/C C++/os_lab_observations$ gedit thread_return_n_5095.c
^C
pv@pv-Vostro-5402:/media/pv/Personal/Code/C C++/os_lab_observations$ gcc thread_return_n_5095.c -lpthread
pv@pv-Vostro-5402:/media/pv/Personal/Code/C C++/os_lab_observations$ ./a.out
Number of students:5
Name 1:Pavana Vamsi
Marks:12 12
Name 2:Vamsi
Marks:34 34
Name 3:Pavana
Marks:11
11
Name 4:chillara v
Marks:50 50
Name 5:sai
Marks:10 10
Name:Pavana Vamsi
Total:24
Name:Vamsi
Total:68
Name:Pavana
Total:22
Name:chillara v
Total:100
Name:sai
Total:20
pv@pv-Vostro-5402:/media/pv/Personal/Code/C C++/os_lab_observations$
```

pv@pv-Vostro-5402: /media/pv/Personal/Code/C C++/os_lab_observations

```
pv@pv-Vostro-5402:/media/pv/Personal/Code/C C++/os_lab_observations$ cat thread_return_n_5095.c
#include<stdio.h>
#include<pthread.h>
struct student
{
char name[20];
int cat1;
int cat2;
int total;
};
void * totalcal(void * st)
{
struct student * s2=(struct student *)st;
s2->total=s2->cat1+s2->cat2;
pthread_exit(NULL);
}
int main()
{
int n;
printf("Number of students:");
scanf("%d",&n);
struct student s[n];
for(int i=0;i<n;i++)
{
printf("Name %d:",i+1);
scanf("\n%[^\n]s",s[i].name);
printf("Marks:");
scanf("%d",&s[i].cat1);
scanf("%d",&s[i].cat2);
s[i].total=0;
}
pthread_t t[n];
for(int i=0;i<n;i++)
{
pthread_create(&t[i],NULL,(void*)totalcal,(void*)&s[i]);
printf("Name:%s\n",s[i].name);
pthread_join(t[i],NULL);
printf("Total:%d\n",s[i].total);
}
return 0;
}

pv@pv-Vostro-5402:/media/pv/Personal/Code/C C++/os_lab_observations$
```

3. Implement CAT1 - thread, process questions

   Hierarchy process creation. Creating a new child process from main and creating the tree using for loops.

pv@pv-Vostro-5402: /media/pv/Personal/Code/C C++/os_lab_observations/lab 5

```
pv@pv-Vostro-5402:/media/pv/Personal/Code/C C++/os_lab_observations/lab 5$ gedit tree_process_5095.c
^C
pv@pv-Vostro-5402:/media/pv/Personal/Code/C C++/os_lab_observations/lab 5$ gcc tree_process_5095.c
pv@pv-Vostro-5402:/media/pv/Personal/Code/C C++/os_lab_observations/lab 5$ ./a.out
L2 child id:15750       its parent in L1 id:15749
L2 child id:15751       its parent in L1 id:15749
L1 child id:15749       its parent in L0 id:15748
L2 child id:15753       its parent in L1 id:15752
L2 child id:15755       its parent in L1 id:15752
L1 child id:15752       its parent in L0 id:15748
L2 child id:15763       its parent in L1 id:15762
L2 child id:15770       its parent in L1 id:15762
L1 child id:15762       its parent in L0 id:15748
L0 the main child process id:15748      it's parent main() id:15747
main() process id:15747
pv@pv-Vostro-5402:/media/pv/Personal/Code/C C++/os_lab_observations/lab 5$
```

pv@pv-Vostro-5402: /media/pv/Personal/Code/C C++/os_lab_observations/lab 5

```
pv@pv-Vostro-5402:/media/pv/Personal/Code/C C++/os_lab_observations/lab 5$ cat tree_process_5095.c
#include <stdio.h>
#include <unistd.h>
int main()
{
int par=fork();
if(par==0)
{
        for (int i = 0; i < 3; i++)
        {
        int l1 = fork();
        if (l1 == 0)
        {
                for (int j = 0; j < 2; j++)
                {
                int l2 = fork();
                if (l2 == 0)
                {
                printf("L2 child id:%d\tits parent in L1 id:%d\n", getpid(), getppid());
                return 0;
                }
                if(l2>0)
                sleep(1);
                }
                printf("L1 child id:%d\tits parent in L0 id:%d\n", getpid(), getppid());
                return 0;
        }
        if(l1>0)
        sleep(4);
        }
        printf("L0 the main child process id:%d\tit\'s parent main() id:%d\n", getpid(), getppid());
        return 0;
}
if(par>0)
{
sleep(15);
printf("main() process id:%d\n",getpid());
}
return 0;
}
pv@pv-Vostro-5402:/media/pv/Personal/Code/C C++/os_lab_observations/lab 5$
```

Hierarchy process creation. Assuming as created process and proceeding the child creation.

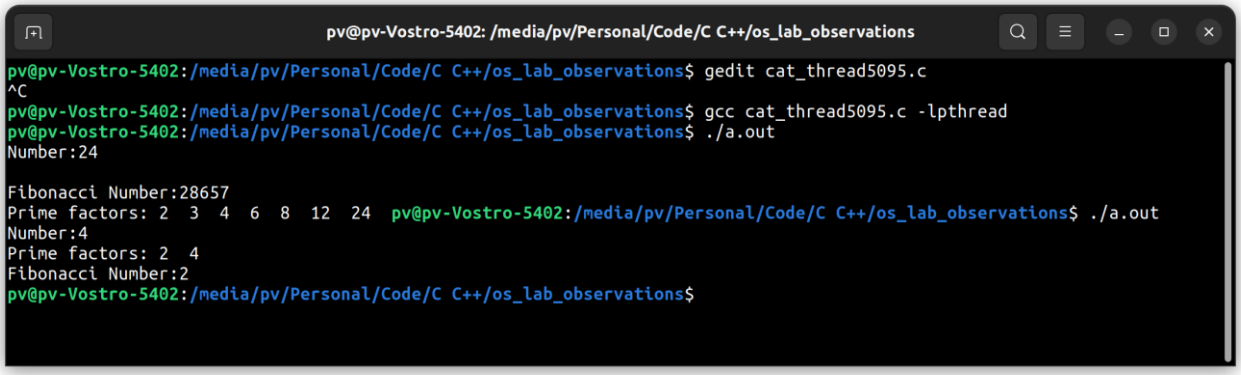
pv@pv-Vostro-5402: /media/pv/Personal/Code/C C++/os_lab_observations/lab 5

```
pv@pv-Vostro-5402:/media/pv/Personal/Code/C C++/os_lab_observations/lab 5$ cat tree_process2_5095.c
#include <stdio.h>
#include <unistd.h>
int main()
{
for (int i = 0; i < 3; i++)
{
        int l1 = fork();
        if (l1 == 0)
        {
                for (int j = 0; j < 2; j++)
                {
                int l2 = fork();
                if (l2 == 0)
                {
                printf("L2 child id:%d\tits parent in L1 id:%d\n", getpid(), getppid());
                return 0;
                }
                if(l2>0)
                sleep(1);
                }
                printf("L1 child id:%d\tits parent in L0 id:%d\n", getpid(), getppid());
                return 0;
        }
        if(l1>0)
        sleep(4);
}
        printf("L0 the main child process id:%d\n", getpid());
return 0;
}
pv@pv-Vostro-5402:/media/pv/Personal/Code/C C++/os_lab_observations/lab 5$
```

```
pv@pv-Vostro-5402: /media/pv/Personal/Code/C C++/os_lab_observations/lab 5
pv@pv-Vostro-5402:/media/pv/Personal/Code/C C++/os_lab_observations/lab 5$ gedit tree_process2_5095.c
^C
pv@pv-Vostro-5402:/media/pv/Personal/Code/C C++/os_lab_observations/lab 5$ gcc tree_process2_5095.c
pv@pv-Vostro-5402:/media/pv/Personal/Code/C C++/os_lab_observations/lab 5$ ./a.out
L2 child id:16122       its parent in L1 id:16121
L2 child id:16123       its parent in L1 id:16121
L1 child id:16121       its parent in L0 id:16120
L2 child id:16127       its parent in L1 id:16126
L2 child id:16128       its parent in L1 id:16126
L1 child id:16126       its parent in L0 id:16120
L2 child id:16138       its parent in L1 id:16137
L2 child id:16139       its parent in L1 id:16137
L1 child id:16137       its parent in L0 id:16120
L0 the main child process id:16120
pv@pv-Vostro-5402:/media/pv/Personal/Code/C C++/os_lab_observations/lab 5$
```

Multi thread program.

```
pv@pv-Vostro-5402: /media/pv/Personal/Code/C C++/os_lab_observations
pv@pv-Vostro-5402:/media/pv/Personal/Code/C C++/os_lab_observations$ cat cat_thread5095.c
#include<stdio.h>
#include<pthread.h>
int primefactor(void* no)
{
        int *n=(int*)no;
        printf("Prime factors: ");
        for(int i=2;i<=*n;i++)
        {
        if(*n%i==0)
        printf("%d  ",i);
        }
        pthread_exit(NULL);
}
void fibonacci(void *no)
{
        int t0=0,t1=1,tn;
        int *n=(int*)no;
        if(*n==0||*n==1)
        printf("\nFibonacci Number:%d\n",*n);
        else
        {
        tn=t0+t1;
        for(int i=3;i<=*n;i++)
        {
        t0=t1;
        t1=tn;
        tn=t0+t1;
        }
        printf("\nFibonacci Number:%d\n",t1);
        }
        pthread_exit(NULL);
}
int main()
{
int n;
printf("Number:");
scanf("%d",&n);
pthread_t t1,t2;
pthread_create(&t1,NULL,(void*)primefactor,&n);
pthread_create(&t2,NULL,(void*)fibonacci,&n);
pthread_join(t1,NULL);
pthread_join(t2,NULL);
return 0;
}
pv@pv-Vostro-5402:/media/pv/Personal/Code/C C++/os_lab_observations$
```


pv@pv-Vostro-5402: /media/pv/Personal/Code/C C++/os_lab_observations
pv@pv-Vostro-5402:/media/pv/Personal/Code/C C++/os_lab_observations$ gedit cat_thread5095.c
^C
pv@pv-Vostro-5402:/media/pv/Personal/Code/C C++/os_lab_observations$ gcc cat_thread5095.c -lpthread
pv@pv-Vostro-5402:/media/pv/Personal/Code/C C++/os_lab_observations$ ./a.out
Number:24
Fibonacci Number:28657
Prime factors: 2 3 4 6 8 12 24 pv@pv-Vostro-5402:/media/pv/Personal/Code/C C++/os_lab_observations$ ./a.out
Number:4
Prime factors: 2 4
Fibonacci Number:2
pv@pv-Vostro-5402:/media/pv/Personal/Code/C C++/os_lab_observations$