TASK 02

```
#include <time.h>

#include <omp.h>

#include <stdio.h>

int main()

{

   struct timespec start, finish;

   double elapsed,elapsed1;

   int size = 20000, i, largest;

   int array[size];

   double openMP,Seq;

   for (int count = 0; count < 6; count++)

   {

       printf("============= Round %d =============", count+1);

       srand(time(NULL));

       for (i = 0; i < size; i++)

       {

           array[i] = ((rand() % 50000)+10000);

       }

       largest = array[0];

       clock_gettime(CLOCK_MONOTONIC, &start);

       for (i = 1; i < size; i++)

       {

           if (largest < array[i])

               largest = array[i];

       }

       clock_gettime(CLOCK_MONOTONIC, &finish);
```

```
        elapsed1 = (finish.tv_sec - start.tv_sec);
        elapsed1 += (finish.tv_nsec - start.tv_nsec) / 1000000000.0;
        printf("\nLargest Element %d\n", largest);
        printf("Execution time [Sequential Computing]:  %f seconds\n",
elapsed1);
        Seq=Seq+elapsed1;
        clock_gettime(CLOCK_MONOTONIC, &start);
#pragma omp parallel for
        for (i = 1; i < size; i++)
        {
            if (largest < array[i])
                largest = array[i];
        }
        clock_gettime(CLOCK_MONOTONIC, &finish);
        elapsed = (finish.tv_sec - start.tv_sec);
        elapsed += (finish.tv_nsec - start.tv_nsec) / 1000000000.0;
        printf("Execution time [OpenMP]:\t\t%f seconds\n\n", elapsed);
        openMP=openMP+elapsed;
        sleep(2);
    }
   printf("\nAverage Sequential Time: %f\nAverage OpenMP time:
%f\n",(Seq/10),(openMP/10));
   return 0;
}
```

```
ammansoomro@Amman-PC:~/Visual_Studio$ ./FinalPrep_Task01
============ Round 1 ============
Largest Element 59998
Execution time [Sequential Computing]:  0.000065 seconds
Execution time [OpenMP]:                0.000063 seconds

============ Round 2 ============
Largest Element 59998
Execution time [Sequential Computing]:  0.000057 seconds
Execution time [OpenMP]:                0.000062 seconds

============ Round 3 ============
Largest Element 59993
Execution time [Sequential Computing]:  0.000057 seconds
Execution time [OpenMP]:                0.000065 seconds

============ Round 4 ============
Largest Element 59997
Execution time [Sequential Computing]:  0.000057 seconds
Execution time [OpenMP]:                0.000059 seconds

============ Round 5 ============
Largest Element 59996
Execution time [Sequential Computing]:  0.000057 seconds
Execution time [OpenMP]:                0.000059 seconds

============ Round 6 ============
Largest Element 59993
Execution time [Sequential Computing]:  0.000057 seconds
Execution time [OpenMP]:                0.000060 seconds


Average Sequential Time: 0.000035
Average OpenMP time: 0.000037
```

TASK 01

```
#include <time.h>

#include <omp.h>

#include <stdio.h>

int main()

{

   struct timespec start, finish;

   double elapsed, elapsed1;

   int size = 100000, i, sum, sum1;

   int array[size];

   double openMP, Seq;

   for (int count = 0; count < 5; count++)

   {

       sum = 0;

       sum1 = 0;

       printf("============= Round %d =============", count + 1);

       srand(time(NULL));

       for (i = 0; i < size; i++)

       {

           array[i] = (rand() % 100);

       }

       clock_gettime(CLOCK_MONOTONIC, &start);

       for (i = 1; i < size; i++)

       {

           sum = sum + array[i];

       }

       clock_gettime(CLOCK_MONOTONIC, &finish);
```

```
        elapsed1 = (finish.tv_sec - start.tv_sec);

        elapsed1 += (finish.tv_nsec - start.tv_nsec) / 1000000000.0;

        printf("\nArray Sum [Sequential Computing]: %d\n", sum);

        printf("Execution time [Sequential Computing]:  %f seconds",
elapsed1);

        Seq = Seq + elapsed1;

        clock_gettime(CLOCK_MONOTONIC, &start);

#pragma omp parallel for

        for (i = 1; i < size; i++)

        {

            sum1 = sum1 + array[i];

        }

        clock_gettime(CLOCK_MONOTONIC, &finish);

        elapsed = (finish.tv_sec - start.tv_sec);

        elapsed += (finish.tv_nsec - start.tv_nsec) / 1000000000.0;

        printf("\nArray Sum [OpenMP]: %d\n", sum1);

        printf("Execution time [OpenMP]:\t\t%f seconds\n\n", elapsed);

        openMP = openMP + elapsed;

        sleep(2);

    }

    printf("\nAverage Sequential Time: %f\nAverage OpenMP time: %f\n",
(Seq / 10), (openMP / 10));

    return 0;

}
```

```
ammansoomro@Amman-PC:~/Visual_Studio$ ./FinalPrep_Task01
============ Round 1 ============
Array Sum [Sequential Computing]: 4952534
Execution time [Sequential Computing]:  0.001587 seconds
Array Sum [OpenMP]: 4952534
Execution time [OpenMP]:                0.001461 seconds

============ Round 2 ============
Array Sum [Sequential Computing]: 4966873
Execution time [Sequential Computing]:  0.000335 seconds
Array Sum [OpenMP]: 4966873
Execution time [OpenMP]:                0.000343 seconds

============ Round 3 ============
Array Sum [Sequential Computing]: 4933202
Execution time [Sequential Computing]:  0.001725 seconds
Array Sum [OpenMP]: 4933202
Execution time [OpenMP]:                0.001394 seconds

============ Round 4 ============
Array Sum [Sequential Computing]: 4953333
Execution time [Sequential Computing]:  0.000353 seconds
Array Sum [OpenMP]: 4953333
Execution time [OpenMP]:                0.000366 seconds

============ Round 5 ============
Array Sum [Sequential Computing]: 4946573
Execution time [Sequential Computing]:  0.000407 seconds
Array Sum [OpenMP]: 4946573
Execution time [OpenMP]:                0.000391 seconds


Average Sequential Time: 0.000441
Average OpenMP time: 0.000395
```

TASK 03

```
#include<stdlib.h>

#include<unistd.h>

#include<pthread.h>

#include<stdio.h>

void *Print (void * x)

{

int a = (int) x;

printf("Hello from thread %u - I was created in iteration %d \n",

pthread_self(), a);

}



int main()

{

int n;

int thread_id;

printf("How many threads are there to be created?\n");

scanf("%d", &n);

pthread_t thread;

thread_id = pthread_self();

for(int i = 1; i <= n; i++){

pthread_create(&thread,NULL,&Print, (void *)(i));

printf("I am thread %u. Created new thread (%u) in iteration
%d\n",thread_id,thread,i);

if((i % 5) == 0)

{

sleep(1);
```

```
}

}

pthread_join(thread,NULL);

//pthread_exit(NULL);

return 0;

}
```

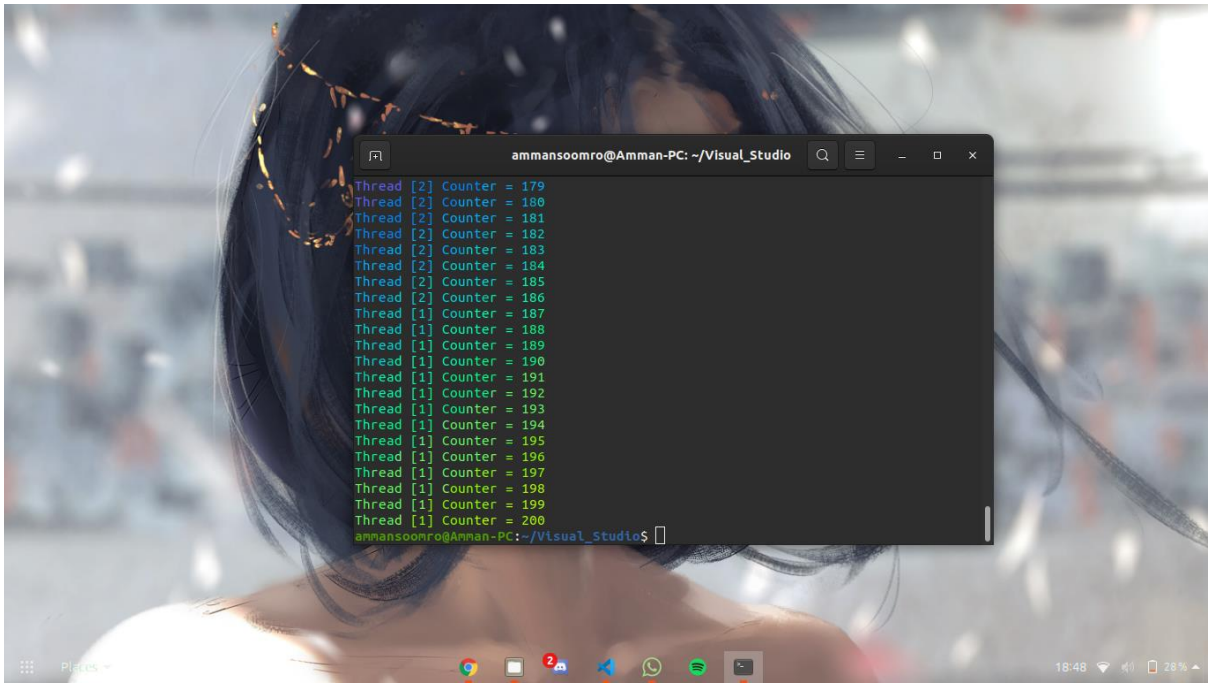
```
ammansoomro@Amman-PC:~/Visual_Studio$ ./FinalPrep_Task03
How many threads are there to be created?
100
I am thread 120923968. Created new thread (120919808) in iteration 1
Hello from thread 120919808 - I was created in iteration 1
I am thread 120923968. Created new thread (112527104) in iteration 2
I am thread 120923968. Created new thread (104134400) in iteration 3
I am thread 120923968. Created new thread (95741696) in iteration 4
I am thread 120923968. Created new thread (87348992) in iteration 5
Hello from thread 87348992 - I was created in iteration 5
Hello from thread 104134400 - I was created in iteration 3
Hello from thread 95741696 - I was created in iteration 4
Hello from thread 112527104 - I was created in iteration 2
```

```
I am thread 120923968. Created new thread (4004644608) in iteration 50
Hello from thread 4004644608 - I was created in iteration 50
I am thread 120923968. Created new thread (3996251904) in iteration 51
I am thread 120923968. Created new thread (3987859200) in iteration 52
I am thread 120923968. Created new thread (3979466496) in iteration 53
I am thread 120923968. Created new thread (3971073792) in iteration 54
I am thread 120923968. Created new thread (3962681088) in iteration 55
```

TASK 04

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


static int counter = 0;
pthread_mutex_t Lock;
void *Increase_Counter(void *number)
{
   int i;
   for (i = 0; i < 100; i++)
   {
       pthread_mutex_lock(&Lock);
       counter++;
       printf("Thread [%d] Counter = %d\n", (int)number, counter);
       pthread_mutex_unlock(&Lock);
   }
}


int main()
{
   int Thread1 = 1, Thread2 = 2;
   pthread_t First_Thread, Second_Thread;
   pthread_mutex_init(&Lock, 0);
   pthread_create(&First_Thread, 0, Increase_Counter, (void *)Thread1);
   pthread_create(&Second_Thread, 0, Increase_Counter, (void
*)Thread2);
   pthread_join(First_Thread, 0);
```

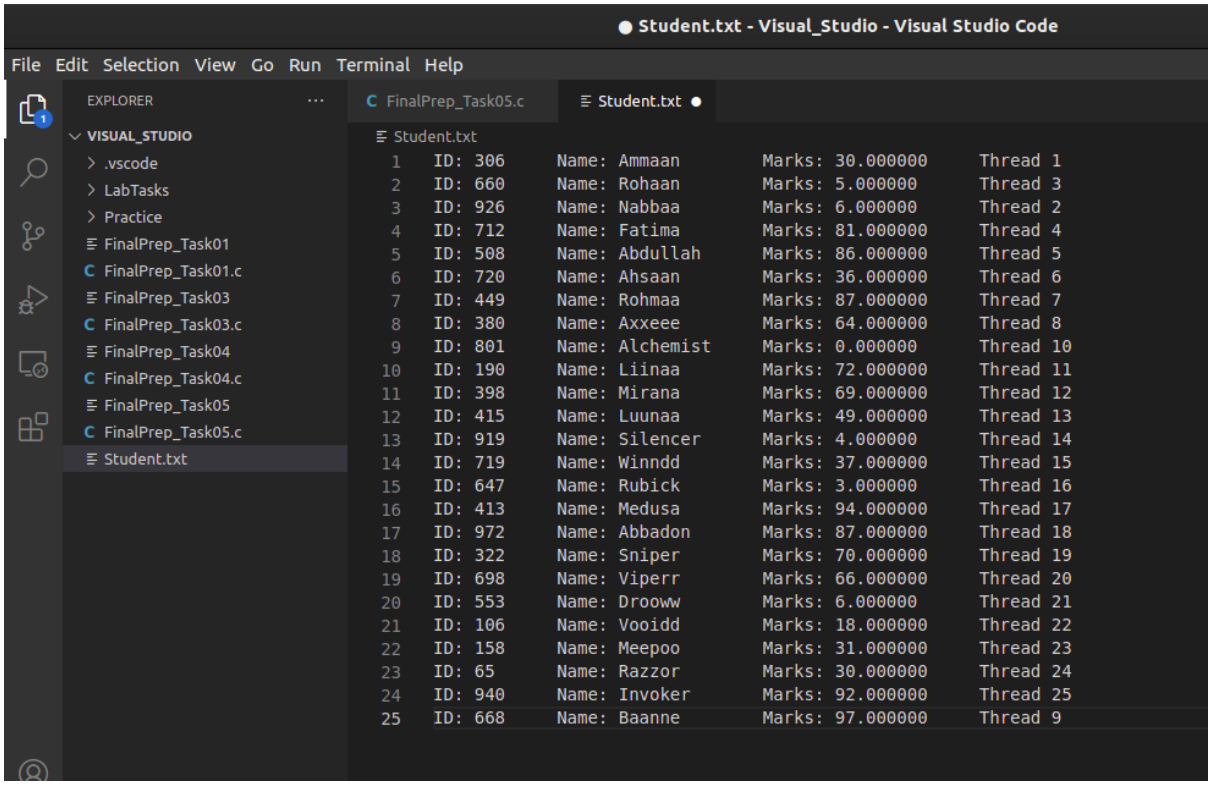
```
    pthread_join(Second_Thread, 0);

    pthread_mutex_destroy(&Lock);

    return 0;

}
```



TASK 06

```c
#include <stdlib.h>

#include <unistd.h>

#include <pthread.h>

#include <time.h>

#include <stdio.h>

struct student {

 char Name[50];

 int ID;

 double Marks;

} s[25];

void *printthread(void *x)

{

   int a = (int)x;

   printf("ID: %d\t", s[a].ID);

   printf("Name: %s\t",s[a].Name);

   printf("Marks: %f\t Thread %d\n",s[a].Marks,a+1);

   sleep(1);

}

int main()

{

   srand(time(NULL));

   for(int i=0; i< 25; i++)

   {

       scanf("%s",s[i].Name);

   }

   for(int i=0; i< 25; i++)
```

```
    {
        s[i].ID = (rand() % 1000);
    }
    for(int i=0; i< 25; i++)
    {
        s[i].Marks = (rand() % 100);
    }
    pthread_t Threads[25];
    for (int i = 0; i < 25; i++)
    {
        pthread_create(&Threads[i], NULL, &printthread, (void *)(i));
    }
    for (int i = 0; i < 25; i++)
    {
        pthread_join(Threads[i], NULL);
    }
    return 0;
}
```


● Student.txt - Visual_Studio - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
VISUAL_STUDIO
.vscode
LabTasks
Practice
FinalPrep_Task01
FinalPrep_Task01.c
FinalPrep_Task03
FinalPrep_Task03.c
FinalPrep_Task04
FinalPrep_Task04.c
FinalPrep_Task05
FinalPrep_Task05.c
Student.txt
FinalPrep_Task05.c
Student.txt ●
Student.txt
1 ID: 306 Name: Ammaan Marks: 30.000000 Thread 1
2 ID: 660 Name: Rohaan Marks: 5.000000 Thread 3
3 ID: 926 Name: Nabbaa Marks: 6.000000 Thread 2
4 ID: 712 Name: Fatima Marks: 81.000000 Thread 4
5 ID: 508 Name: Abdullah Marks: 86.000000 Thread 5
6 ID: 720 Name: Ahsaan Marks: 36.000000 Thread 6
7 ID: 449 Name: Rohmaa Marks: 87.000000 Thread 7
8 ID: 380 Name: Axxeee Marks: 64.000000 Thread 8
9 ID: 801 Name: Alchemist Marks: 0.000000 Thread 10
10 ID: 190 Name: Liinaa Marks: 72.000000 Thread 11
11 ID: 398 Name: Mirana Marks: 69.000000 Thread 12
12 ID: 415 Name: Luunaa Marks: 49.000000 Thread 13
13 ID: 919 Name: Silencer Marks: 4.000000 Thread 14
14 ID: 719 Name: Winndd Marks: 37.000000 Thread 15
15 ID: 647 Name: Rubick Marks: 3.000000 Thread 16
16 ID: 413 Name: Medusa Marks: 94.000000 Thread 17
17 ID: 972 Name: Abbadon Marks: 87.000000 Thread 18
18 ID: 322 Name: Sniper Marks: 70.000000 Thread 19
19 ID: 698 Name: Viperr Marks: 66.000000 Thread 20
20 ID: 553 Name: Drooww Marks: 6.000000 Thread 21
21 ID: 106 Name: Vooidd Marks: 18.000000 Thread 22
22 ID: 158 Name: Meepoo Marks: 31.000000 Thread 23
23 ID: 65 Name: Razzor Marks: 30.000000 Thread 24
24 ID: 940 Name: Invoker Marks: 92.000000 Thread 25
25 ID: 668 Name: Baanne Marks: 97.000000 Thread 9