

Programme Name: \_\_\_\_\_\_\_\_**BCS HONS**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Course Code: \_\_**CSC 2624**\_\_\_\_\_\_\_\_

Course Name: \_\_\_\_\_\_\_**Distributed And Parallel Computing**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Assignment** / Lab Sheet / Project / Case Study No. \_**1**\_\_\_

Date of Submission: \_\_\_\_\_\_**7/12/2021**\_\_\_\_\_\_\_\_\_\_\_\_\_

**Submitted By: Submitted To:**

Student Name**: Dipesh Tha Shrestha** Faculty Name**: Manoj Gautam**

IUKL ID: **041902900028** Department**: LMS**

Semester**: Fourth Semester**

Intake**: September 2019**

1. **Write a C++ thread based program that implements FOUR (4) threads such that each thread run FOUR (4) different functions add(int a, int b), sub(int a, int b), divide(int a, int b) and multiply(int a, int b). And waits for the main thread before terminating and also implement MUTEX to avoid the race condition for IO.**

**Code:**

#include<bits/stdc++.h>

using **namespace** std;

**int** ans=0;

mutex mu;

**void** print(**int** result)

{

    mu.lock();

cout<<result<<endl;

 mu.unlock();

}

**void** add(**int** a, **int** b)

{

**int** ans=a+b;

  print(ans);

}

**void** sub(**int** a, **int** b)

{

**int** ans=a-b;

 print(ans);

}

**void** multiply(**int** a, **int** b)

{

**int** ans=a\*b;

 print(ans);

}

**void** divide(**int** a, **int** b)

{

**int** ans=a/b;

 print(ans);

}

**int** main(**int** argc, **char** **const** **\***argv[])

{

**int** a,b;

    cin>>a>>b;

    thread t1(add,a,b);

 thread t2(sub,a,b);

 thread t3(divide,a,b);

thread t4(multiply,a,b);

    t1.join();

t2.join();

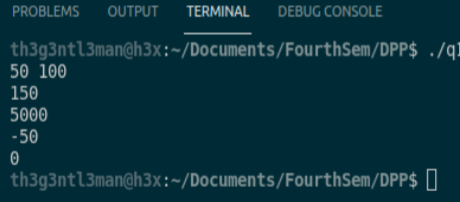
 t3.join();

 t4.join();

    return 0;

}

Output:



1. **Write a C++ threaded program to compute the sum of 10 Billon (10000000000) natural number using For loop. You should split the For loops into FOUR (4) different threads and compute the sum independently. Use Main thread to compute the total sum from four different thread and print the result.**

**Hint: Use shared memory variable to access the result of child threads from main threads.**

**Code:**

#include<bits/stdc++.h>

typedef **unsigned** **long** **long** **int** ull ;

using **namespace** std;

*//ull sum=0;*

ull bigSum=0;

mutex mu;

**void** print(ull result)

{

    mu.lock();

    bigSum=bigSum+result;

    mu.unlock();

}

**void** loop1()

{

    ull sum1=0;

for(ull i=1;i<250000000;i++)

    {

        sum1=sum1+i;

    }

   print(sum1);

}

**void** loop2()

{

    ull sum2=0;

 for(ull i=2500000000;i<5000000000;i++)

    {

        sum2=sum2+i;

    }

    print(sum2);

}

**void** loop3()

{

    ull sum3=0;

 for(ull i=5000000000;i<7500000000;i++)

    {

        sum3=sum3+i;

    }

    print(sum3);

}

**void** loop4()

{

    ull sum4=0;

  for(ull i=7500000000;i<=10000000000;i++)

    {

        sum4=sum4+i;

    }

    print(sum4);

}

**int** main(**int** argc, **char** **const** **\***argv[])

{

    thread t1(loop1);

 thread t2(loop2);

 thread t3(loop3);

 thread t4(loop4);

    t1.join();

  t2.join();

 t3.join();

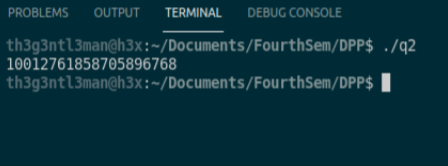
 t4.join();

    cout<<bigSum<<endl;

    return 0;

}

**Output:**

****

**Thank you**