1. Informations about computer system used:

CPU: Intel® Core™ i5-8300H

NUMBER OF PHYSICAL CORRES: 4

NUMBER OF LOGICAL CORRES: 8

CLOCK RATE: 2300 - 4000 MHz

CPU CACHE: 8 MB

2. Source code:

```
#include <iostream>
#include <string>
#include <thread>
#include <windows.h>
#include <mutex>
#include <chrono>
#include <vector>
#include <atomic>
using namespace std;

static int global_variable = 0;
static const int max_global_variable = 10000000;
static atomic<int> atomic_global_variable;

mutex mtx;

//FIRST TASK
class FunctionObject
{
public:
void operator()(string text)
{
cout << "Using FunctionObject is : " << this_thread::get_id() << " " << text << endl;
}
};
class Class
{
public:
static void Method(string text)
{
cout << "Using ClassMethod is : " << this_thread::get_id() << " " << text << endl;
}
};
void GlobalFunction(string text)
{
cout << "Using GlobalFunction is : " << this_thread::get_id() << " " << text << endl;
}
void First()
{
cout << "FIRST TASK" << endl;
thread First(GlobalFunction, "thread running.");
Sleep(100); //I used Sleep functions only in order to get more readable output.
thread Second(FunctionObject(), "thread running.");
Sleep(100);
thread Third(Class::Method, "thread running.");
Sleep(100);
thread Fourth([](string text) {
cout << "Using LambdaFunction is : " << this_thread::get_id() << " " << text << endl; }, "thread
running.");
```

```
First.join();
Second.join();
Third.join();
Fourth.join();
}

//SECOND TASK
void PrintSomeText(string text)
{
for (int i = 0; i < 50; i++)
{
lock_guard<mutex> lock(mtx);
cout << "Print number: " << i + 1 << " from: "<< this_thread::get_id() <<" "<< text << endl;
}
}
void Second()
{
cout << "SECOND TASK" << endl;
static const int thread_number = 20;
thread T[thread_number];
for (int i = 0; i < thread_number; i++)
{
T[i] = thread(PrintSomeText, "thread.");
}
for (int i = 0; i < thread_number; i++)
{
T[i].join();
}
}

//THIRD TASK
struct Timer
{
chrono::time_point<chrono::steady_clock>start, end;
chrono::duration<float> duration;

Timer()
{
start = chrono::high_resolution_clock::now();
}
~Timer()
{
end = chrono::high_resolution_clock::now();
chrono::duration<float>duration = end - start;
float ms = duration.count() * 1000.0f;
cout << ms <<" ms" << endl;
}
};
void CounterFunction(int t)
{
if (t == 0)
{
for (int i = 0; i < max_global_variable; i++)
{
global_variable++;
}
}
if (t == 1)
{
for (int i = 0; i < max_global_variable; i++)
{
lock_guard<mutex> lock(mtx);
global_variable++;
}
}
if (t == 2)
{
for (int i = 0; i < max_global_variable; i++)
{
atomic_global_variable++;
}
}
}
void OneTimeExecution()
{
Timer time;
global_variable = 0;
return CounterFunction(0);
}
void ThreadsExecution(int n)
{
Timer time;
global_variable = 0;
const int thread_vector_number = 10;
vector<thread> thread_vector;
thread_vector.reserve(thread_vector_number);
for (int i = 0; i < thread_vector_number; i++)
{
thread_vector.emplace_back(thread(CounterFunction,n));
}
for (auto& entry : thread_vector)
{
entry.join();
}
}
void Third()
{
cout << "THIRD TASK" << endl;


cout << "Time elapsed for one time execution of function is : ";
OneTimeExecution();
cout << "Global variable value for one time execution is :" << global_variable << endl;

cout << "Time elapsed for 10 unsynchronized threads : ";
ThreadsExecution(0);
cout << "Global variable value for unsychronized incrementation :" << global_variable << endl;

cout << "Time elapsed for 10 threads using mutex is : ";
ThreadsExecution(1);
cout << "Global variable value using mutex is :" << global_variable << endl;

cout << "Time elapsed for 10 threads using atomic variable is : ";
ThreadsExecution(2);
cout << "Global variable value using atomic variable is :" << atomic_global_variable << endl;
}

int main()
{
First();
Second();
Third();
return 0;
}
```

3. **Times report for sub-task 3.**

Times for debug mode:

1. Time elapsed for one time execution of function is : **17.9356 ms**

   Global variable value for one time execution is :**10000000**

2. Time elapsed for 10 unsynchronized threads : **305.225 ms**

   Global variable value for unsychronized incrementation :**16653410**

3. Time elapsed for 10 threads using mutex is : **18446.6 ms**

   Global variable value using mutex is :**100000000**

4. Time elapsed for 10 threads using atomic variable is : **1834 ms**

   Global variable value using atomic variable is :**100000000**

Times for relase mode:

1. Time elapsed for one time execution of function is : **1.639 ms**

   Global variable value for one time execution is :**10000000**

2. Time elapsed for 10 unsynchronized threads : **12.2626 ms**

   Global variable value for unsychronized incrementation :**20000000**

3. Time elapsed for 10 threads using mutex is : **2401.12 ms**

   Global variable value using mutex is :**100000000**

4. Time elapsed for 10 threads using atomic variable is : **1681.21 ms**

   Global variable value using atomic variable is :**100000000**

4. **Briefly comment for sub-task 3.**

- Relase mode brings significant changes execution time of program.
- The most significant time duration changes can be seen for threads using mutex.
- Not achieving value of 100000000 for global variable using unsychronized variant was caused by race condition of threads.
- Mutex and atomic variables allow multiple threads to avoid race condition by restricting access of multiple threads to shared data.