Thread

Example 1 - Simple thread
Create simple_thread.cpp and write the following code:

```cpp
#include <iostream>
#include <thread>
#include <unistd.h>

using namespace std;

void hello()
{
    sleep(1);
    cout<<"Hello World from thread\n";
}

int main(int argc, char const *argv[])
{
    cout<<"Before thread \n";
    thread t1(hello);
    t1.join();
    cout<<"After thread \n";
    return 0;
}

// compile and run thread programs by:
g++ simple_thread.cpp -lpthread
./a.out
```

Example 2 - Sum with two thread
Create sum.cpp and the copy/paste the following code to sum up 1 to n.

```cpp
#include <iostream>

using namespace std;

int main(int argc, char ** argv) {
    unsigned long number = atol(argv[1]);
    long sum = 0;
    for (long i = 0; i <= number; i++)
        sum += i;
    cout << "Sum = " << sum << endl;
    return (0);
```
Compile and run the code by:
- g++ sum.cpp -o sum
- time ./simple_sum 1,000,000,000

Now, we try to rewrite the above example with threads. Create thread_sum.cpp and write the following code:

```cpp
#include <iostream>
#include <thread>
#include <unistd.h>

using namespace std;
long total = 0;

void sum(long first, long end){
    long s=0;
    for (long i=first; i<=end; i++)
        s+=i;
    total +=s;
}

int main(int argc, char const *argv[]){
    long number = atol(argv[1]);
    thread t1(sum, 0, number/2);
    thread t2(sum, (number/2)+1, number);
    t1.join();
    t2.join();
    cout<<"Sum = "<<total;
    return 0;
}
```

Compile and run the code by:
- Compile and run the code by:
g++ thread_sum.cpp -o thread_sum
time ./thread_sum 1,000,000,000

- Which code is faster? Why?
- How many threads did we use?

**Thread ID**

Create the thread_id.cpp and write the following code:
#include <iostream>
#include<thread>
#include<unistd.h>

using namespace std;

void hello()
{
    cout<<"Hello 
";
    cout << this_thread::get_id() << '
';
}

int main(int argc, char const *argv[]) {
    thread t1(hello);
    t1.join();

    cout<<"Possible Thread: "<<thread::hardware_concurrency()<<endl;

    return 0;
}

- What is your output? Discuss it with your peer.

---

Array of threads

Crate the multithread.cpp and write the following code:

```cpp
#include<iostream>
#include <thread>

using namespace std;

void hello(int i) {
    cout << "Hello from thread # " << i << 'n';
}

int main(int argc, char const *argv[]) {
    thread t[10];
    for (int i=0; i<10; i++){
        t[i] = thread(hello, i);
    }

    for (int i=0; i<10; i++){
        t[i].join();
    }
    return 0;
}
Compare your output with your peer? Is it differences? Discuss it with your peer.
How you can solve Question 4 of the assignment? What is the biggest challenge?