Challenge-14

```
#include <iostream>

#include <vector>

#include <cuda_runtime.h>


__global__

void fibonacci_kernel(unsigned long long *fib, int N)

{

  int i = blockIdx.x * blockDim.x + threadIdx.x;

  if (i < N)

  {

    if (i == 0) fib[i] = 0;

    else if (i == 1) fib[i] = 1;

    else

    {

      unsigned long long a = 0, b = 1, c;

      for (int j = 2; j <= i; ++j)

      {

        c = a + b;

        a = b;

        b = c;

      }

      fib[i] = b;

    }

  }

}


void fibonacci_cpu(std::vector<unsigned long long>& fib, int N)

{

  fib[0] = 0;
```

```
    fib[1] = 1;
    for (int i = 2; i < N; ++i)
        fib[i] = fib[i-1] + fib[i-2];
}

int main()
{
    const int N = 1 << 20; // 2^20 ~ 1 million numbers
    std::vector<unsigned long long> fib_cpu(N);
    std::vector<unsigned long long> fib_gpu(N);

    // CPU computation
    cudaEvent_t start_cpu, stop_cpu;
    cudaEventCreate(&start_cpu);
    cudaEventCreate(&stop_cpu);
    cudaEventRecord(start_cpu);

    fibonacci_cpu(fib_cpu, N);

    cudaEventRecord(stop_cpu);
    cudaEventSynchronize(stop_cpu);

    float ms_cpu = 0;
    cudaEventElapsedTime(&ms_cpu, start_cpu, stop_cpu);

    // GPU computation
    unsigned long long *d_fib;
    cudaMalloc(&d_fib, N * sizeof(unsigned long long));

    cudaEvent_t start_gpu, stop_gpu;
    cudaEventCreate(&start_gpu);
    cudaEventCreate(&stop_gpu);
```

```
cudaEventRecord(start_gpu);


int blockSize = 256;
int numBlocks = (N + blockSize - 1) / blockSize;
fibonacci_kernel<<<numBlocks, blockSize>>>(d_fib, N);
cudaMemcpy(fib_gpu.data(), d_fib, N * sizeof(unsigned long long), cudaMemcpyDeviceToHost);


cudaEventRecord(stop_gpu);
cudaEventSynchronize(stop_gpu);


float ms_gpu = 0;
cudaEventElapsedTime(&ms_gpu, start_gpu, stop_gpu);


// Validate results
bool correct = true;
for (int i = 0; i < N; ++i)
{
  if (fib_cpu[i] != fib_gpu[i])
  {
    correct = false;
    std::cout << "Mismatch at index " << i << ": CPU=" << fib_cpu[i] << ", GPU=" << fib_gpu[i] << "\n";
    break;
  }
}


if (correct)
  std::cout << "✅ Results match!\n";
else
  std::cout << "❌ Results mismatch!\n";


std::cout << "CPU time: " << ms_cpu << " ms\n";
std::cout << "GPU time: " << ms_gpu << " ms\n";
```

```
    cudaFree(d_fib);


    return 0;
}
```