Hands-on Activity 13.1	
Parallelism and Multithreading	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed:** November 11,2025
Section: CPE21S4	**Date Submitted:** November 11, 2025
Name(s): Bautista, Mariela S.	**Instructor:** Engr. Jimlord Quejado

6. Output

ILO A and ILO B:

Screenshot

CMakeLists.txt × CPE010_HOA13.1_Bautista_sc1.cpp ×

```cpp
#include <iostream>
#include <thread>
using namespace std;

void print(int n, const string &str) {
    cout << "Printing integer: " << n << endl;
    cout << "Printing string: " << str << endl;
}

int main() {
    thread t1([>>]print, 10, "T.I.P.");
    t1.join();
    return 0;
}
```

Output:

```
C:\Users\Mariela\CLionProjects\Parallelism_and_Multithreading\cmake-build-debug\Parallelism_and_Multithreading.exe
Printing integer: 10
Printing string: T.I.P.

Process finished with exit code 0
```

Analysis

In this part, I learned how to create a single thread. I declared a function named print() that takes an integer and a string, then printed both. By using std::thread, I created a thread t1 that runs this function The join() command ensures that the main thread waits for t1 to finish before continuing, preventing errors. This helped me figure out how one thread can perform a specific function independently.

Table 13-1. Simple One-Threaded Example

Screenshot	CMakeLists.txt CPE010_HOA13.1_BAUTISTA_sc3.cpp CPE010_HOA13.1_BAUTISTA_sc2.cpp **Output:**
Analysis	I used multiple threads to run the same function at the same time. Each thread printed a different message, which made the execution faster and more efficient.I learned that threads can work together to share tasks and use the CPU better. The join() function made sure each thread finished before the program ended. This helped me understand how useful multithreading is for doing many tasks at once.

```cpp
#include <iostream>
#include <thread>
#include <vector>
using namespace std;

void print(int n, const string &str) {
    string msg = to_string(n) + " : " + str;
    cout << msg << endl;
}

int main() {
    vector<string> s = {"T.I.P.", "Competent", "Computer", "Engineers"};
    vector<thread> threads;

    for (int i = 0; i < s.size(); i++) {
        threads.push_back(thread(print, i, s[i]));
    }

    for (auto &th : threads) {
        th.join();
    }
    return 0;
}
```

```
C:\Users\Mariela\CLionProjects\Parallelism_and_Multithreading\cmake-build-debug\Parallelism_and_Multithreading.exe
0 : T.I.P.1 : Competent
2 : Computer
3 : Engineers
```

Table 13-2. Multithreaded Example

7. Supplementary Activity

Part A: Demonstrate an understanding in Parallelism, Concurrency, and Multithreading in C++ by answering the given questions. Use of supplementary materials to support answers must be cited as reference.

Questions:

1. **Write a definition of multithreading and its advantages/disadvantages.**
   Multithreading is the ability of the program to run multiple threads at once to perform different tasks simultaneously. Its main advantage is faster execution by using multiple CPU cores efficiently. However, it can lead to issues like race conditions and complex debugging when threads share data.

2. **Rationalize the use of multithreading by providing at least 3 use-cases.**
   1st, is running multiple background tasks like downloads or file reading.
   2nd, is Handling client requests in servers
   3rd, is Performing parallel computations in scientific or gaming applications.

3. **Difference between parallelism and concurrency.**
   Parallelism means performing multiple tasks exactly at the same time on different cores, while concurrency means managing multiple tasks that appear to run together by switching between them efficiently.

Part B: Create C++ Code and show a solution that satisfies the given requirements below.

Screenshot of Code

CMakeLists.txt | CPE010_HOA13.1_sc3.cpp

```cpp
#include <iostream>
#include <thread>
using namespace std;

int globalVar = 0;

void add(int value) {
    globalVar += value;
    cout << "Added " << value << ", GlobalVar = " << globalVar << endl;
}

int main() {
    thread t1([>>]add, 5);
    thread t2([>>]add, 10);
    thread t3([>>]add, 20);

    cout << "Before joining threads, GlobalVar = " << globalVar << endl;
    t1.join();
    cout << "After T1.join(), GlobalVar = " << globalVar << endl;
    t2.join();
    cout << "After T2.join(), GlobalVar = " << globalVar << endl;
    t3.join();
    cout << "After T3.join(), GlobalVar = " << globalVar << endl;

    return 0;
```

Output

```
C:\Users\Mariela\CLionProjects\PM\cmake-build-debug\PM.exe
Before joining threads, GlobalVar = Added 5, GlobalVar = 35Added 10, GlobalVar = 35
35
Added 20, GlobalVar = 35

After T1.join(), GlobalVar = 35
After T2.join(), GlobalVar = 35
After T3.join(), GlobalVar = 35

Process finished with exit code 0
```

Analysis	When I tried running the program, I noticed the global variable's value sometimes changed in unexpected ways. It made me see how threads can affect each other when they run together. Using join() helped fix that and made the program run smoother. This part showed me how timing matters when using multiple threads.

Part C: Use a multithreading with one of the algorithms previously developed in the course; provide an analysis of the result.

Screenshot of Code

```cpp
#include <iostream>
#include <thread>
#include <vector>
using namespace std;

int result1 = -1, result2 = -1;

void searchPart(const vector<int>& arr, int start, int end, int target, int& result) {
    for (int i = start; i < end; i++) {
        if (arr[i] == target) {
            result = i;
            break;
        }
    }
}

int main() {
    vector<int> arr = {5, 12, 8, 20, 7, 15, 3, 10, 25, 18};
    int target = 15;

    int mid = arr.size() / 2;

    thread t1( f: [>>] searchPart, cref(arr), 0, [>>] mid, [>>] target, ref( [&] result1));
    thread t2( f: [>>] searchPart, cref(arr), [>>] mid, arr.size(), [>>] target, ref( [&] result2));

    t1.join();
    t2.join();

    if (result1 != -1)
        cout << "Element found at index (thread 1): " << result1 << endl;
    else if (result2 != -1)
        cout << "Element found at index (thread 2): " << result2 << endl;
    else
        cout << "Element not found" << endl;

    return 0;
}
```

Output

```
C:\Users\Mariela\CLionProjects\PM\cmake-build-debug\PM.exe
Element found at index (thread 2): 5

Process finished with exit code 0
```

Analysis	I used multithreading to speed up a simple linear search. I divided the array into two halves and searched each half at the same time using two threads. This helped me see how multiple threads can work together to find data faster. It was interesting to observe that both threads ran independently but still shared the same goal. I learned that even a simple algorithm like searching can become faster and more efficient when done in parallel.

8. Conclusion

Through this activity, I learned how parallelism and multithreading can make programs faster by allowing different parts to run at the same time. I also understood how important it is to control how threads work together to avoid errors when sharing data. By trying both single-threaded and multi-threaded examples, I saw the difference between running one task at a time and running several at once, and how using join() helps keep everything in order. The supplementary tasks, like using global variables and doing searches with threads, helped me apply what I learned in a more practical way. Overall, I think I did pretty well in this hands-on-activity because I managed to make the programs work and understand their behavior, but I still want to improve at managing thread timing and avoiding race conditions. This activity really showed me how powerful and useful multithreading can be when used properly.

9. Assessment Rubric

Rubric for SO 7 (7)

Criteria	Ratings						Pts
SO 7 PI 1 IILO4 Utilize lifelong learning skills in pursuit of personal development and excellence in professional practice. threshold: 4.8 pts	6 pts Excellent | Educational interests and pursuits exist and flourish outside classroom requirements,knowledge and/or experiences are pursued independently and applies knowledge learned into practice	5 pts Good | Educational interests and pursuits exist and flourish outside classroom requirements,knowledge and/or experiences are pursued independently	4 pts Satisfactory | Look beyond classroom requirements, showing interest in pursuing knowledge independently	3 pts Unsatisfactory | Begins to look beyond classroom requirements, showing interest in pursuing knowledge independently	2 pts Poor | Relies on classroom instruction only	1 pts Very Poor | No initiative or interest in acquiring new knowledge	6 pts
SO 7 PI 2 IILO4 Utilize lifelong learning skills in pursuit of personal development and excellence in professional practice. threshold: 4.8 pts	6 pts Excellent | Completes an assigned task independently and practices continuous improvement	5 pts Good | Completes an assigned task without supervision or guidance	4 pts Satisfactory | Requires minimal guidance to complete an assigned task	3 pts Unsatisfactory | Requires detailed or step-by-step instructions to complete a task	2 pts Poor | Shows little interest to complete a task independently	1 pts Very Poor | No interest to complete a task independently	6 pts
SO 7 PI 3 IILO4 Utilize lifelong learning skills in pursuit of personal development and excellence in professional practice. threshold: 4.8 pts	6 pts Excellent | Synthesizes and integrates information from a variety of sources; formulates a clear and precise perspective; draws appropriate conclusions	5 pts Good | Evaluate information from a variety of sources; formulates a clear and precise perspective.	4 pts Satisfactory | Analyze information from a variety of sources; formulates a clear and precise perspective.	3 pts Unsatisfactory | Apply the gathered information to formulate the problem	2 pts Poor | Gather and summarized the information from a variety of sources but failed to formulate the problem	1 pts Very Poor | Gather information from a variety of sources	6 pts
SO 7 PI 4 IILO4 Utilize lifelong learning skills in pursuit of personal development and excellence in professional practice. threshold: 4.8 pts	6 pts Excellent | Ideas are combined in original and creative ways in line with the new and emerging technology trends to solve a problem or address an issue.	5 pts Good | Ideas are creative and adapt the new knowledge to solve a problem or address an issue	4 pts Satisfactory | Ideas are creative in solving a problem, or address an issue	3 pts Unsatisfactory | Shows some creative ways to solve the problem	2 pts Poor | Shows initiative and attempt to develop creative ideas to solve the problem	1 pts Very Poor | Ideas are copied or restated from the sources consulted	6 pts

Total Points: 24

10. References

- **GeeksforGeeks. *"Multithreading in C++."* Retrieved from:** https://www.geeksforgeeks.org/multithreading-in-cpp/

- **GeeksforGeeks. *"Thread Synchronization in C++."* Retrieved from:** https://www.geeksforgeeks.org/thread-synchronization-in-cpp/

- **GeeksforGeeks. *"C++ Program for Linear Search."* Retrieved from https://www.geeksforgeeks.org/linear-search/**

- **Tip Learning Platform. (n.d.). *File 29597219* [PDF file]. In Course 66762. Canvas. https://tip.instructure.com/courses/66762/files/29597219?wrap=1**