Date: _____ M T W T F S S

NAME : JAWAD AHMED
ROLL NO : 20P-0165
Section : BCS-6A

Assignment # 03

Subject: Parallel and Distributed Computing
Date: 6 Feburary, 2023

Q:- What are Detachable and Joinable Threads?

Ans: ⊛ Joinable Threads: Joinable threads are threads in a multithreaded environment that can be joined by other threads.

→ Joinable thread has a specific end point and produces a result that can be waited for or obtained by other thread.

→ Joinable threads are useful to synchronize multiple threads and coordinate their execution.

→ When thread calls a 'join()' method calling Thread will be blocked until the joined thread finishes it's execution.

P.T.O

Date:

CPP Code For Joinable Thread:

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

void joinable_thread_example(int count) {
        while( count --> 0)
                cout << "Hello" <<endl;
        std::this_thread::sleep_for(chrono::seconds(5));
}

int main() {
        std::thread t1(joinable_thread_example, 2);
        cout << "Before Join" <<endl;
        t1.join();
        cout << "After Join" <<endl;
        return 0;
}
```

Output:

1. Before Join
2. Hello
3. Hello
4. After Join

Code Explanation: In this example the thread t1 is joined then the main thread execution stopped and wait for t1 to complete and after that main thread completed it's execution. In the output the "After Join" printed after t1 termination.

P.T.O

Date: M T W T F S S

Detachable Threads: Detachable thread also known as non-joinable threads.

→ Detachable threads run independently and cannot be joined by other threads.

→ Run in background and donot block. execution of other threads.

→ Useful for the tasks that donot need to be synchronized with the main thread or other threads, such as Background tasks or cleanup operations.

→ A detached thread does-not have a specific end point and doesnot produce a result that can be waited for or obtained by another thread.

CPP Code for Detachable Thread

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

void detachable_thread_example(int count) {
        while (count --> 0)
                cout << "Hello" <<endl;
        std::this_thread::sleep_for(chrono::seconds(5));
}
```


P.T.O

Date: _____ M T W T F S S

```
int main() {

    std::thread t1(detachable_thread_example, 2);
    cout << "Before Join" << endl;
    t1.detach();
    cout << "After Join" << endl;
    return 0;
}
```

Output:

Before Join
After Join

Explanation: Now In this example you notice that thread t1 is not executed is because we have detached the thread and before the turn of t1 comes the main thread terminates. So as the main thread terminates all other threads also terminated.

But In case of join the main thread by waited for t1 to completely executed and then proceeded next.