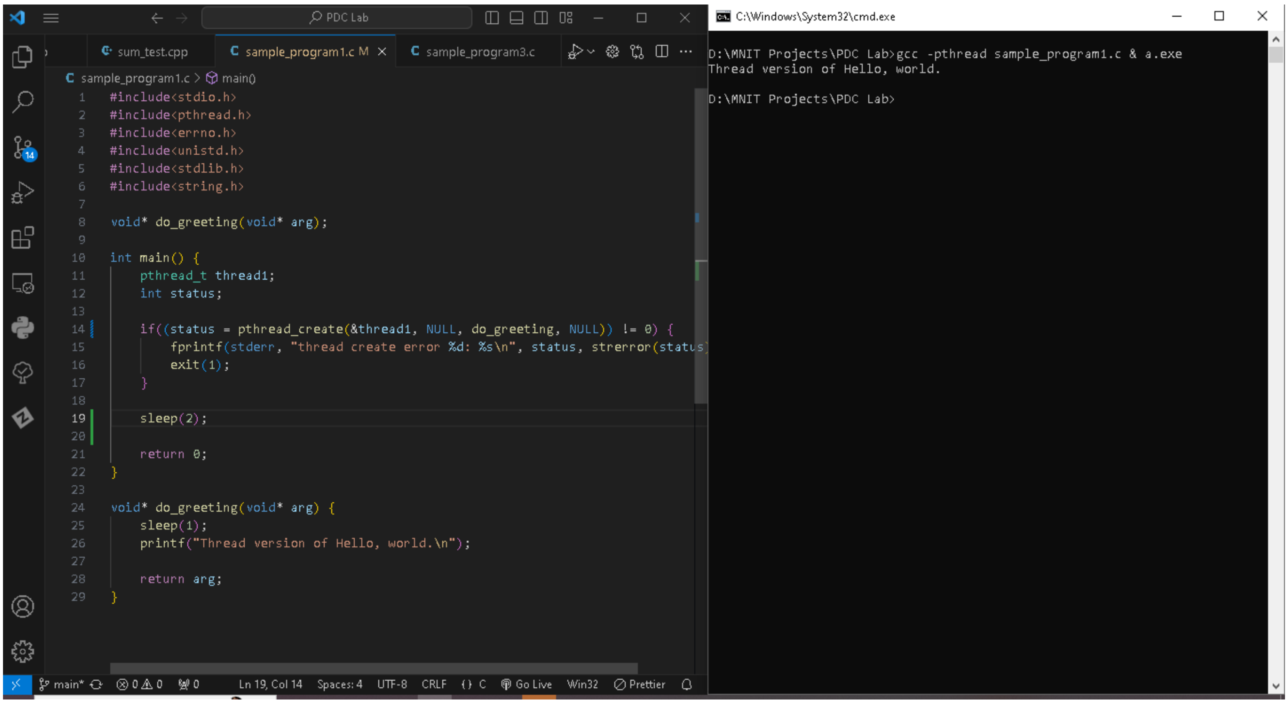
Name: Ranjan Baro

ID: 2023PCP5274

1. Describe/explain your observations, i.e. what must have happened in the original, unmodified Sample 1 program?

Ans. When original program is run, we do not observe any output or print statement. After inserting a **2 seconds *sleep* ( )** into the ***main*** function (after thread creation), we can observe the following output.



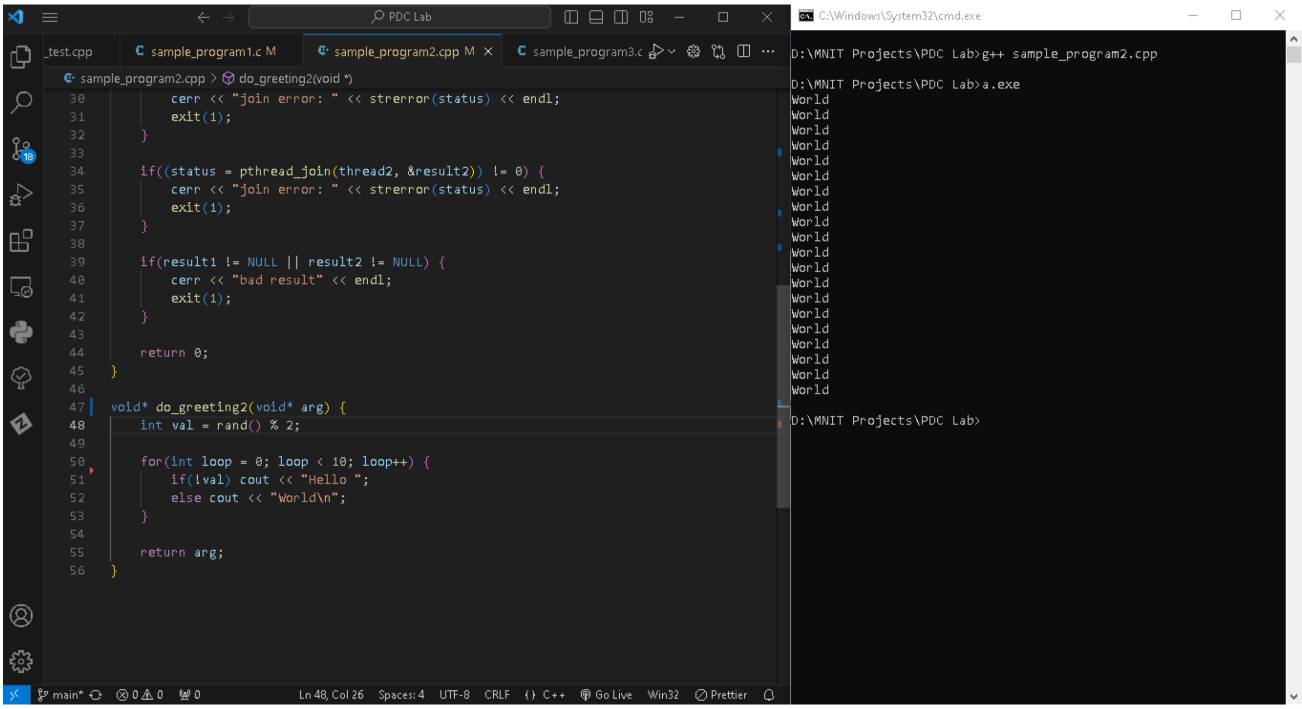
```c
#include<stdio.h>
#include<pthread.h>
#include<errno.h>
#include<unistd.h>
#include<stdlib.h>
#include<string.h>

void* do_greeting(void* arg);

int main() {
    pthread_t thread1;
    int status;

    if((status = pthread_create(&thread1, NULL, do_greeting, NULL)) != 0) {
        fprintf(stderr, "thread create error %d: %s\n", status, strerror(status)
        exit(1);
    }

    sleep(2);

    return 0;
}

void* do_greeting(void* arg) {
    sleep(1);
    printf("Thread version of Hello, world.\n");

    return arg;
}
```

```
D:\MNIT Projects\PDC Lab>gcc -pthread sample_program1.c & a.exe
Thread version of Hello, world.

D:\MNIT Projects\PDC Lab>
```

We have not observed any output in original program, because the main thread and the created thread run concurrently and the program may exit before the created thread complete its execution of ***do_greeting*** function. When we insert **2 seconds *sleep*** in the ***main*** function, the main program waits for 2 seconds before exiting and during this wait, the created thread will complete its execution of ***do_greeting*** function and show the print statement of the ***do_greeting*** function.

2. Report your results, particularly the observed formatting of the sample 2 program.

Ans. Sample program 2 generates two thread and both threads execute ***do_greeting2*** function. The two threads run concurrently and may print **Hello** or **World** depending on the generated random variable **val**. We may observe the following output.

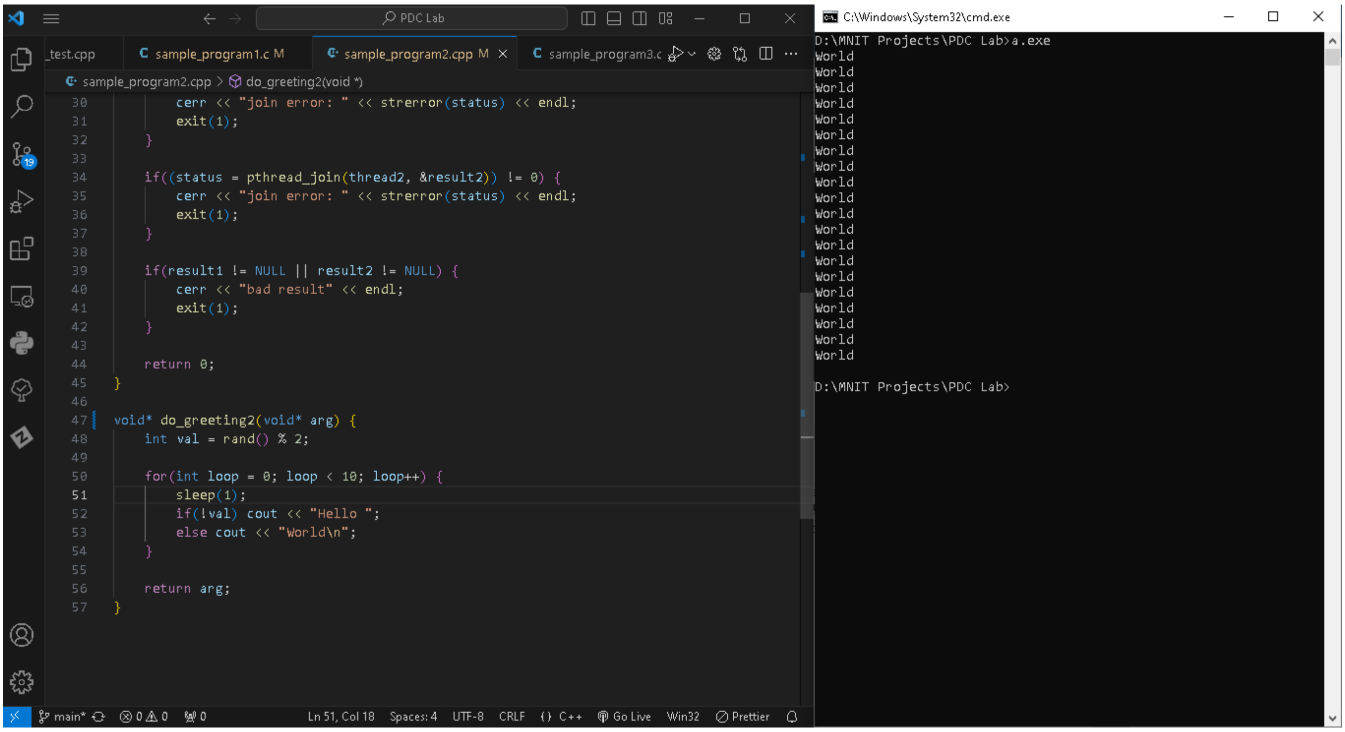

PDC Lab
_test.cpp
sample_program1.c M
sample_program2.cpp M
sample_program3.c
sample_program2.cpp > do_greeting2(void *)
C:\Windows\System32\cmd.exe
main* 0 0 0 Ln 48, Col 26 Spaces: 4 UTF-8 CRLF C++ Go Live Win32 Prettier

```cpp
        cerr << "join error: " << strerror(status) << endl;
        exit(1);
    }

    if((status = pthread_join(thread2, &result2)) != 0) {
        cerr << "join error: " << strerror(status) << endl;
        exit(1);
    }

    if(result1 != NULL || result2 != NULL) {
        cerr << "bad result" << endl;
        exit(1);
    }

    return 0;
}

void* do_greeting2(void* arg) {
    int val = rand() % 2;

    for(int loop = 0; loop < 10; loop++) {
        if(!val) cout << "Hello ";
        else cout << "World\n";
    }

    return arg;
}
```

```
D:\MNIT Projects\PDC Lab>g++ sample_program2.cpp

D:\MNIT Projects\PDC Lab>a.exe
World
World
World
World
World
World
World
World
World
World
World
World
World
World
World
World
World
World
World
World

D:\MNIT Projects\PDC Lab>
```

3. Report your results again. Explain why they are different from the results seen in question 2.

Ans. When we insert **one second *sleep* ( )** at the beginning of the loop in the ***do_greeting2* ( )** function, we see delay in the execution of threads. There is one second delay between each print statement **Hello** or **World**. Here is the output.

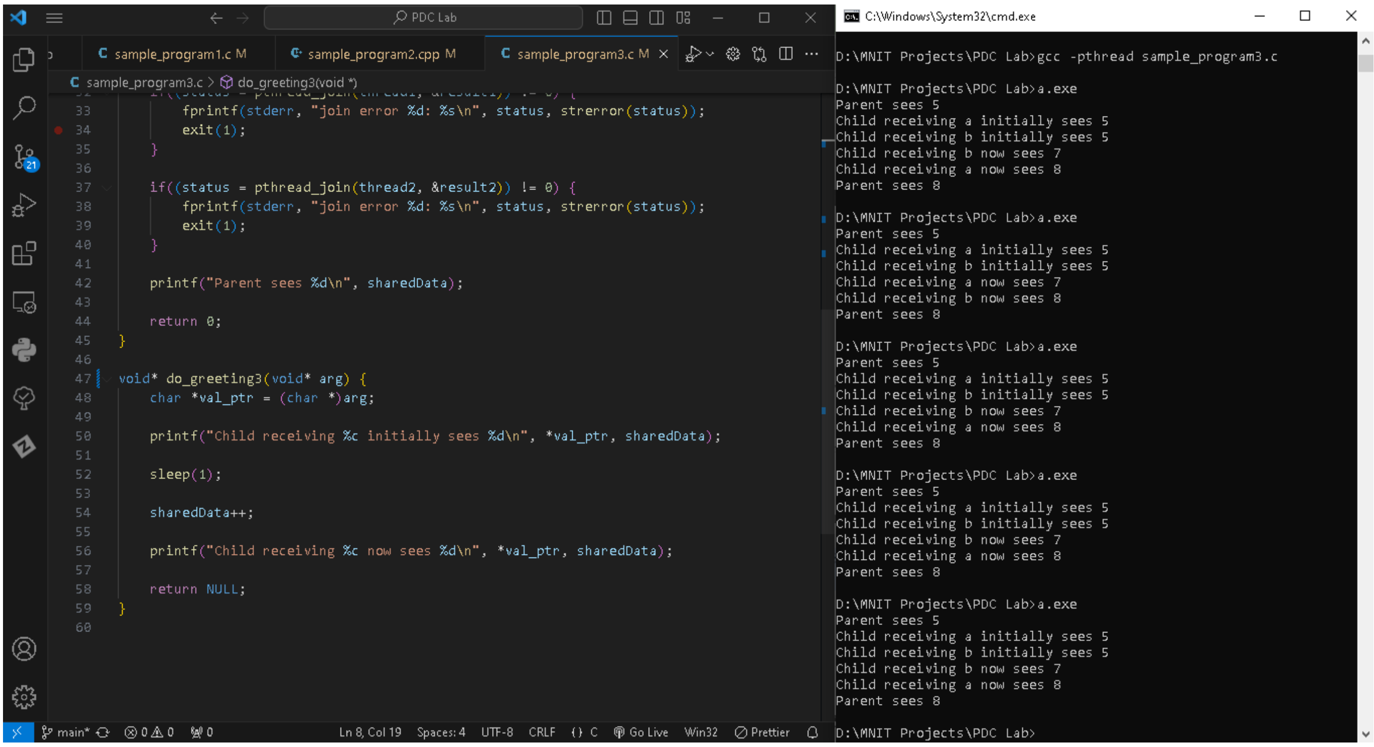

PDC Lab
_test.cpp
sample_program1.c M
sample_program2.cpp M
sample_program3.c
sample_program2.cpp > do_greeting2(void *)
C:\Windows\System32\cmd.exe
main* 0 0 0 Ln 51, Col 18 Spaces: 4 UTF-8 CRLF C++ Go Live Win32 Prettier

```cpp
        cerr << "join error: " << strerror(status) << endl;
        exit(1);
    }

    if((status = pthread_join(thread2, &result2)) != 0) {
        cerr << "join error: " << strerror(status) << endl;
        exit(1);
    }

    if(result1 != NULL || result2 != NULL) {
        cerr << "bad result" << endl;
        exit(1);
    }

    return 0;
}

void* do_greeting2(void* arg) {
    int val = rand() % 2;

    for(int loop = 0; loop < 10; loop++) {
        sleep(1);
        if(!val) cout << "Hello ";
        else cout << "World\n";
    }

    return arg;
}
```

```
D:\MNIT Projects\PDC Lab>a.exe
World
World
World
World
World
World
World
World
World
World
World
World
World
World
World
World
World
World
World
World

D:\MNIT Projects\PDC Lab>
```

5. Compile the sample program 3 and run it multiple times (you may see some variation between runs). Choose one particular sample run. Describe, trace, and explain the output of the program.

Ans. In sample 3 program, the **main** thread creates two additional threads. The main thread prints **sharedData** and other two thread also print **sharedData** along with char **val** (**a** or **b**). Here is the output of the program.



```c
        fprintf(stderr, "join error %d: %s\n", status, strerror(status));
        exit(1);
    }

    if((status = pthread_join(thread2, &result2)) != 0) {
        fprintf(stderr, "join error %d: %s\n", status, strerror(status));
        exit(1);
    }

    printf("Parent sees %d\n", sharedData);

    return 0;
}

void* do_greeting3(void* arg) {
    char *val_ptr = (char *)arg;

    printf("Child receiving %c initially sees %d\n", *val_ptr, sharedData);

    sleep(1);

    sharedData++;

    printf("Child receiving %c now sees %d\n", *val_ptr, sharedData);

    return NULL;
}
```

```
D:\MNIT Projects\PDC Lab>gcc -pthread sample_program3.c

D:\MNIT Projects\PDC Lab>a.exe
Parent sees 5
Child receiving a initially sees 5
Child receiving b initially sees 5
Child receiving b now sees 7
Child receiving a now sees 8
Parent sees 8

D:\MNIT Projects\PDC Lab>a.exe
Parent sees 5
Child receiving a initially sees 5
Child receiving b initially sees 5
Child receiving a now sees 7
Child receiving b now sees 8
Parent sees 8

D:\MNIT Projects\PDC Lab>a.exe
Parent sees 5
Child receiving a initially sees 5
Child receiving b initially sees 5
Child receiving b now sees 7
Child receiving a now sees 8
Parent sees 8

D:\MNIT Projects\PDC Lab>a.exe
Parent sees 5
Child receiving a initially sees 5
Child receiving b initially sees 5
Child receiving b now sees 7
Child receiving a now sees 8
Parent sees 8

D:\MNIT Projects\PDC Lab>a.exe
Parent sees 5
Child receiving a initially sees 5
Child receiving b initially sees 5
Child receiving b now sees 7
Child receiving a now sees 8
Parent sees 8

D:\MNIT Projects\PDC Lab>
```

Execution steps of the program -

- **i)** Main thread creates two threads which execute do_greeting3 function passing the character value **a** or **b**. After creating two threads, it prints **sharedData**'s value i.e. **Parent sees 5.**
- **ii)** The two children thread start executing ***do_greeting3*** function. Each child thread then prints **sharedData**.
- **iii)** After 1 second sleep, each child updates **sharedData**'s value and prints the s**haredData**'s value.
- **iv)** At last, the main thread prints the updated value of **sharedData**.

6. Explain in your own words how the thread-specific (not shared) data is communicated to the child threads.

Ans. Each thread in a multi thread program have a local copy of a variable. This variable can be updated without interfering with other thread's variable. Main thread can set and get values for this variable. This allows it to communicate initial values or any updates to the child thread.