

NAME : Jawad Ahmed  
RollNO : 20P-0165  
Section : BCS-4A

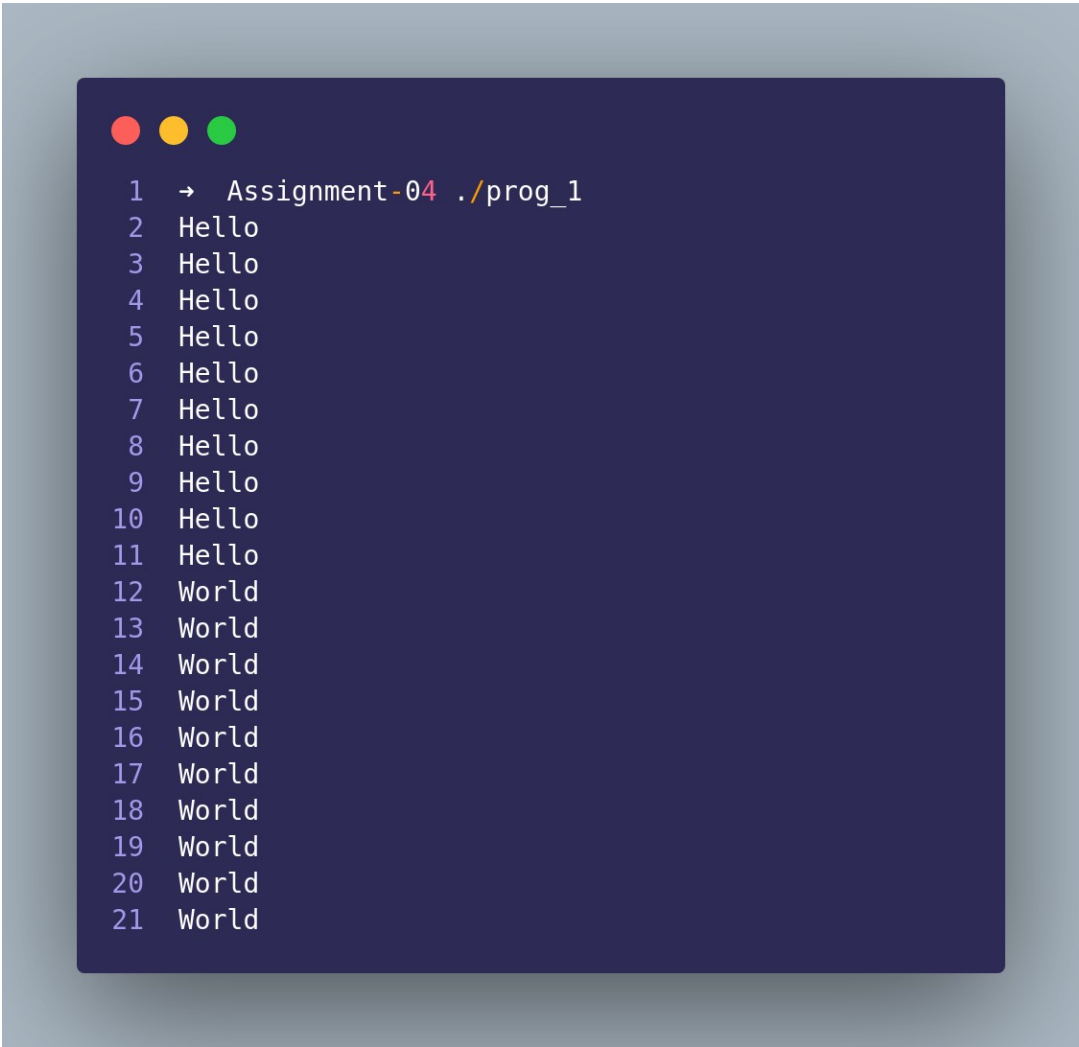
1:

```
1  #include <stdio.h>
2  #include <pthread.h>
3  #include <stdlib.h>
4
5  void *thread1()
6  {
7      for (int c = 0; c < 10; c++)
8      {
9          printf("Hello \n");
10     }
11 }
12
13 void *thread2()
14 {
15     for (int c = 0; c < 10; c++)
16     {
17         printf("World \n");
18     }
19 }
20
21 int main()
22 {
23     int status;
24     pthread_t tid1, tid2;
25
26     pthread_create(&tid1, NULL, thread1, NULL);
27     pthread_create(&tid2, NULL, thread2, NULL);
28
29     pthread_join(tid1, NULL);
30     pthread_join(tid2, NULL);
31
32     return 0;
33 }
```

2:

```
1  → Assignment-04 gcc -pthread -o prog_1 prog_1.c
2
```

### 3: OUTPUT

A terminal window with a dark blue background and light blue text. It shows the execution of a program named 'prog\_1'. The output consists of 21 lines, numbered 1 to 21. Lines 1 through 10 output 'Hello', and lines 11 through 21 output 'World'. The prompt 'Assignment-04' is visible at the top of the terminal.

```
1 → Assignment-04 ./prog_1
2 Hello
3 Hello
4 Hello
5 Hello
6 Hello
7 Hello
8 Hello
9 Hello
10 Hello
11 Hello
12 World
13 World
14 World
15 World
16 World
17 World
18 World
19 World
20 World
21 World
```

### 4: Explanation of the OUTPUT:

In this program what we have did is that we have created the two threads that will run two functions one function name is **thread1** and the other function name is **thread2**. *The first thread will be going to run the thread1 function now it can happen that during the running of the thread1 the preemption can come and the control pass to second thread but in my case the first thread executed completely and after that the second thread runs and it completed it's execution and program exits. We have used the pthread\_join that mean that don't exit the program until the given thread complete it's execution. If you don't add that the main exits and we all know that when main exits our program also exits. Also in the parameters when calling the function we passed **NULL** that mean that we are not passing that argument.*

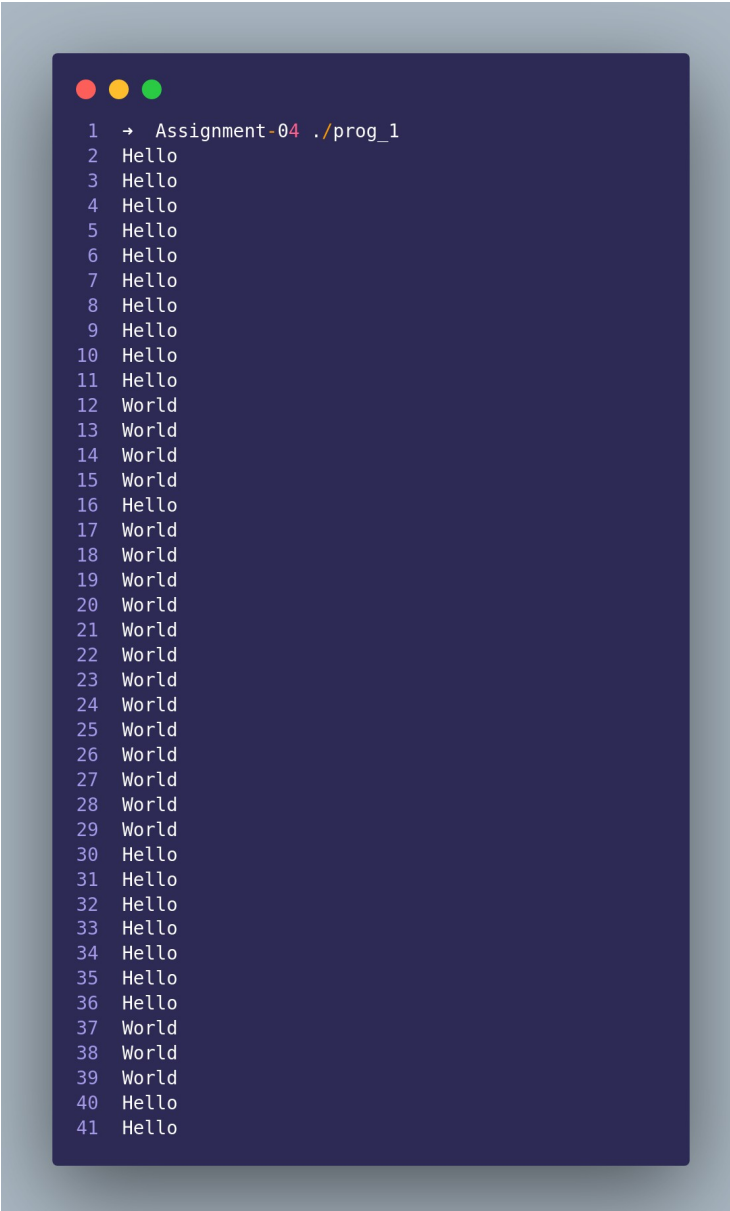
5:

```
1  #include <stdio.h>
2  #include <pthread.h>
3  #include <stdlib.h>
4
5  void *thread1()
6  {
7      for (int c = 0; c < 10; c++)
8      {
9          printf("Hello \n");
10     }
11 }
12
13 void *thread2()
14 {
15     for (int c = 0; c < 10; c++)
16     {
17         printf("World \n");
18     }
19 }
20
21 int main()
22 {
23     int status;
24     pthread_t tid1, tid2, tid3, tid4;
25
26     pthread_create(&tid1, NULL, thread1, NULL);
27     pthread_create(&tid2, NULL, thread2, NULL);
28     pthread_create(&tid3, NULL, thread1, NULL);
29     pthread_create(&tid4, NULL, thread2, NULL);
30
31     pthread_join(tid1, NULL);
32     pthread_join(tid2, NULL);
33     pthread_join(tid3, NULL);
34     pthread_join(tid4, NULL);
35
36     return 0;
37 }
38
```

6:

```
1
2 → Assignment-04 gcc -pthread -o prog_1 prog_1.c
3
4
```

7:

A terminal window with a dark blue background and light blue text. The window has three colored window control buttons (red, yellow, green) in the top-left corner. The text inside the terminal shows a sequence of 41 lines, numbered 1 to 41 on the left. The output consists of 'Hello' and 'World' strings. The first 10 lines are 'Hello', followed by 10 'World's, then another 10 'Hello's, and finally 11 'World's. This interleaving demonstrates thread execution and CPU scheduling.

```
1  → Assignment-04 ./prog_1
2  Hello
3  Hello
4  Hello
5  Hello
6  Hello
7  Hello
8  Hello
9  Hello
10 Hello
11 Hello
12 World
13 World
14 World
15 World
16 Hello
17 World
18 World
19 World
20 World
21 World
22 World
23 World
24 World
25 World
26 World
27 World
28 World
29 World
30 Hello
31 Hello
32 Hello
33 Hello
34 Hello
35 Hello
36 Hello
37 World
38 World
39 World
40 Hello
41 Hello
```

***Now In this case the preemption occurs and the output of different threads shown in the way CPU schedule the threads or Light Weight Process.***

***(a) Compile and execute the program.***

```
1  /* Includes */
2  #include <unistd.h> /* Symbolic Constants */
3  #include <sys/types.h> /* Primitive System Data Types */
4  #include <errno.h> /* Errors */
5  #include <stdio.h> /* Input/Output */
6  #include <stdlib.h> /* General Utilities */
7  #include <pthread.h> /* POSIX Threads */
8  #include <string.h> /* String handling */
9
10 #define NUM_RUNS 10000000
11
12 /* prototype for thread routine */
13 void handler(void *ptr);
14
15 int counter; /* shared variable */
16
17 int main()
18 {
19     int i[2];
20     pthread_t thread_a;
21     pthread_t thread_b;
22
23     i[0] = 0; /* argument to threads */
24     i[1] = 1;
25
26     pthread_create(&thread_a, NULL, (void *)&handler, (void *)&i[0]);
27     pthread_create(&thread_b, NULL, (void *)&handler, (void *)&i[1]);
28
29     pthread_join(thread_a, NULL);
30     pthread_join(thread_b, NULL);
31
32     printf("-----\n");
33     printf("Final Counter value: %d\n", counter);
34     printf("Error: %d\n", (NUM_RUNS * 2 - counter));
35     exit(0);
36 }
37
38 void handler(void *ptr)
39 {
40     int iter = 0;
41     int thread_num;
42     thread_num = *((int *)ptr);
43     printf("Starting Thread %d \n", thread_num);
44
45     while (iter < NUM_RUNS)
46     {
47         counter++;
48         iter += 1;
49     }
50     printf("Thread %d, counter = %d \n", thread_num, counter);
51     pthread_exit(0); /* exit thread */
52 }
```

***Answer the following questions:***

***1: What should be the value of the counter variable at the end?***

***Ans:  $2 * \text{NUM\_RUNS}$  or  $2 * 10000000 \Rightarrow 20000000$***

***2: What is the value you get?***

***Ans: Counter Value  $\Rightarrow 10357845$***

***3: How large is the error and how much does it vary on different runs?***

***Ans: Error  $\Rightarrow 9642155$  , The Value of the Error Lies between 8000000 and 10000000 and it not greater then this range nor less than.***

***4: How much user time (rougly) does the program take to run on your system?***

***Ans: By Taking the Average User Time  $\Rightarrow 0.80\text{sec}$***