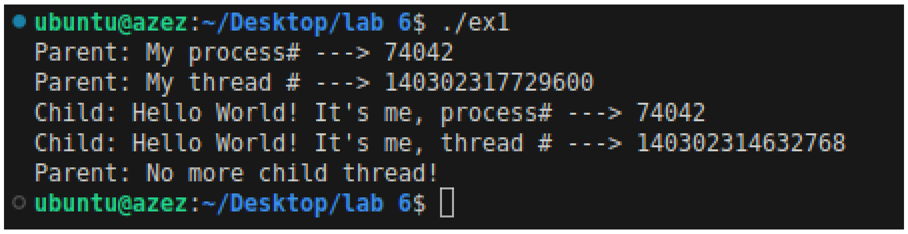
Name	Abdulaziz Hamid Ebrahim
ID	1946282

Ex1:

1) Run the above program, and observe its output:



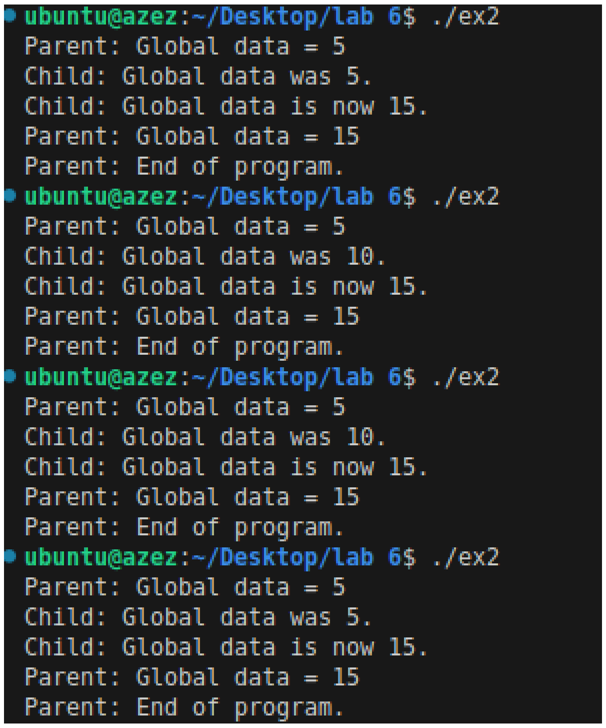
```
ubuntu@azez:~/Desktop/lab 6$ ./ex1
Parent: My process# ---> 74042
Parent: My thread # ---> 140302317729600
Child: Hello World! It's me, process# ---> 74042
Child: Hello World! It's me, thread # ---> 140302314632768
Parent: No more child thread!
ubuntu@azez:~/Desktop/lab 6$
```

2) Are the process ID numbers of parent and child threads the same or different? Why?

No, they are not. We know that every thread has its own unique ID, and the program's output confirms that.

Ex2:

3) Run the above program several times; observe its output every time. A sample output follows:



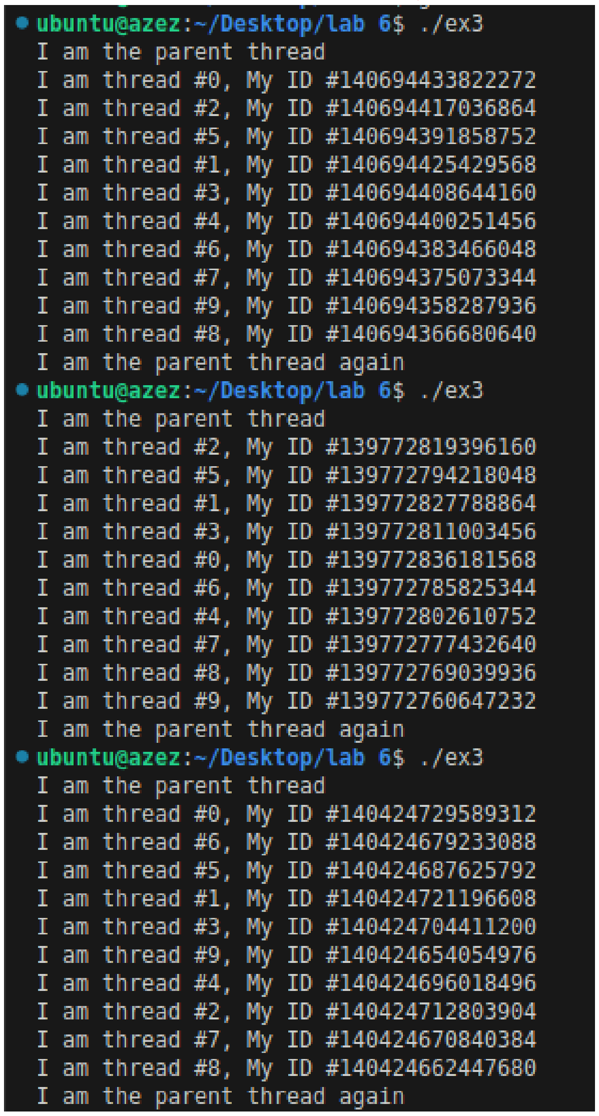
```
ubuntu@azez:~/Desktop/lab 6$ ./ex2
Parent: Global data = 5
Child: Global data was 5.
Child: Global data is now 15.
Parent: Global data = 15
Parent: End of program.
ubuntu@azez:~/Desktop/lab 6$ ./ex2
Parent: Global data = 5
Child: Global data was 10.
Child: Global data is now 15.
Parent: Global data = 15
Parent: End of program.
ubuntu@azez:~/Desktop/lab 6$ ./ex2
Parent: Global data = 5
Child: Global data was 10.
Child: Global data is now 15.
Parent: Global data = 15
Parent: End of program.
ubuntu@azez:~/Desktop/lab 6$ ./ex2
Parent: Global data = 5
Child: Global data was 5.
Child: Global data is now 15.
Parent: Global data = 15
Parent: End of program.
```

4) Does the program give the same output every time? Why?
Yes, the program's output is not guaranteed to be the same every time. It seems to be the same in this instance because my system completed the thread's job before the main thread could modify glob_data. However, this is not a guaranteed outcome on every system or every run.

5) Do the threads have separate copies of glob_data?
No, Threads do not have separate copies of glob_data. They share memory.

Ex3:

6) Run the above program several times and observe the outputs:



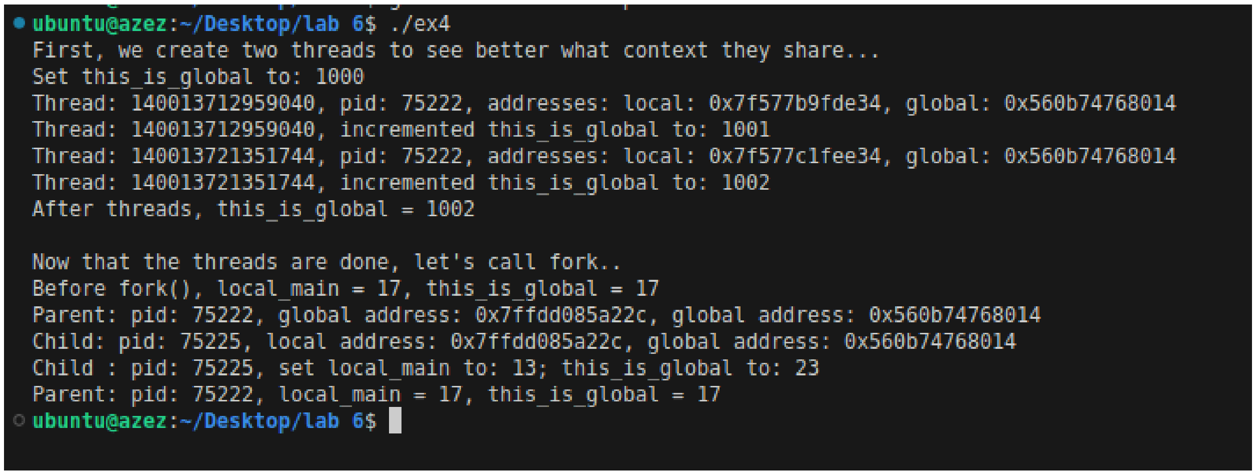
```
ubuntu@azez:~/Desktop/lab 6$ ./ex3
I am the parent thread
I am thread #0, My ID #140694433822272
I am thread #2, My ID #140694417036864
I am thread #5, My ID #140694391858752
I am thread #1, My ID #140694425429568
I am thread #3, My ID #140694408644160
I am thread #4, My ID #140694400251456
I am thread #6, My ID #140694383466048
I am thread #7, My ID #140694375073344
I am thread #9, My ID #140694358287936
I am thread #8, My ID #140694366680640
I am the parent thread again
ubuntu@azez:~/Desktop/lab 6$ ./ex3
I am the parent thread
I am thread #2, My ID #139772819396160
I am thread #5, My ID #139772794218048
I am thread #1, My ID #139772827788864
I am thread #3, My ID #139772811003456
I am thread #0, My ID #139772836181568
I am thread #6, My ID #139772785825344
I am thread #4, My ID #139772802610752
I am thread #7, My ID #139772777432640
I am thread #8, My ID #139772769039936
I am thread #9, My ID #139772760647232
I am the parent thread again
ubuntu@azez:~/Desktop/lab 6$ ./ex3
I am the parent thread
I am thread #0, My ID #140424729589312
I am thread #6, My ID #140424679233088
I am thread #5, My ID #140424687625792
I am thread #1, My ID #140424721196608
I am thread #3, My ID #140424704411200
I am thread #9, My ID #140424654054976
I am thread #4, My ID #140424696018496
I am thread #2, My ID #140424712803904
I am thread #7, My ID #140424670840384
I am thread #8, My ID #140424662447680
I am the parent thread again
```

7) Do the output lines come in the same order every time? Why?

No, the output lines do not appear in the same order each time. This variability results from differences in thread creation times and priorities, which change with every run.

Ex4:

8) Run the above program and observe its output. Following is a sample output:



```
● ubuntu@azez:~/Desktop/lab 6$ ./ex4
  First, we create two threads to see better what context they share...
  Set this_is_global to: 1000
  Thread: 140013712959040, pid: 75222, addresses: local: 0x7f577b9fde34, global: 0x560b74768014
  Thread: 140013712959040, incremented this_is_global to: 1001
  Thread: 140013721351744, pid: 75222, addresses: local: 0x7f577c1fee34, global: 0x560b74768014
  Thread: 140013721351744, incremented this_is_global to: 1002
  After threads, this_is_global = 1002

  Now that the threads are done, let's call fork..
  Before fork(), local_main = 17, this_is_global = 17
  Parent: pid: 75222, global address: 0x7ffdd085a22c, global address: 0x560b74768014
  Child: pid: 75225, local address: 0x7ffdd085a22c, global address: 0x560b74768014
  Child : pid: 75225, set local_main to: 13; this_is_global to: 23
  Parent: pid: 75222, local_main = 17, this_is_global = 17
○ ubuntu@azez:~/Desktop/lab 6$ 
```

9) Did this_is_global change after the threads have finished? Why?

Yes, the value of this_is_global does change after the threads have finished. This change occurs because it was modified after calling the join() function for all threads..

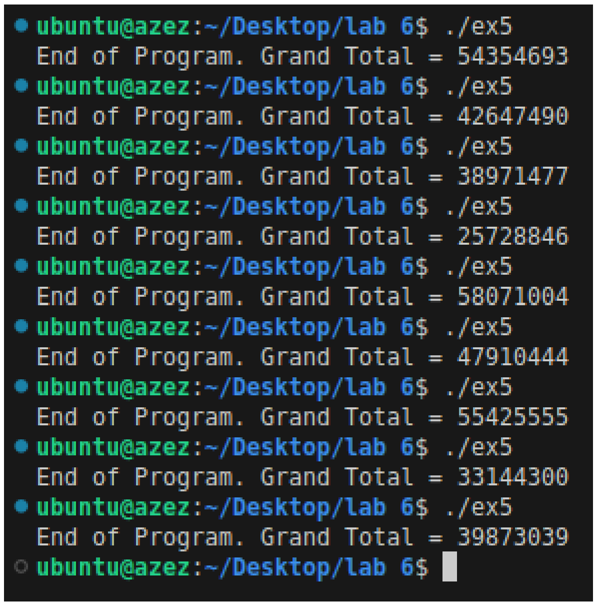
10) Are the local addresses the same in each thread? What about the global addresses?
Each thread has a unique local address. However, global addresses are identical across threads.

11) Did local_main and this_is_global change after the child process has finished? Why?
Yes, the values of local_main and this_is_global change after the child process has finished because the parent can modify these values after the child completes its task.

12) Are the local addresses the same in each process? What about global addresses? What happened?
Yes, in each process, the local addresses can be the same because processes do not share the same memory space and thus have separate memory addresses. This difference in memory allocation explains why processes can have the same local addresses but not global ones

Ex5:

13) Run the above program several times and observe the outputs, until you get different results.



```
ubuntu@azez:~/Desktop/lab 6$ ./ex5
End of Program. Grand Total = 54354693
ubuntu@azez:~/Desktop/lab 6$ ./ex5
End of Program. Grand Total = 42647490
ubuntu@azez:~/Desktop/lab 6$ ./ex5
End of Program. Grand Total = 38971477
ubuntu@azez:~/Desktop/lab 6$ ./ex5
End of Program. Grand Total = 25728846
ubuntu@azez:~/Desktop/lab 6$ ./ex5
End of Program. Grand Total = 58071004
ubuntu@azez:~/Desktop/lab 6$ ./ex5
End of Program. Grand Total = 47910444
ubuntu@azez:~/Desktop/lab 6$ ./ex5
End of Program. Grand Total = 55425555
ubuntu@azez:~/Desktop/lab 6$ ./ex5
End of Program. Grand Total = 33144300
ubuntu@azez:~/Desktop/lab 6$ ./ex5
End of Program. Grand Total = 39873039
ubuntu@azez:~/Desktop/lab 6$
```

14) How many times the line tot_items = tot_items + *iptr; is executed?

The line "tot_items = tot_items + *iptr" is executed 2,500,000 times. This count results from having 50 threads each running the command 50,000 times.

15) What values does *iptr have during these executions?

Points to the data's value in the tidrec struct for the thread
In other word, *iptr points to the data's value in the tidrec struct for the current thread

16) What do you expect Grand Total to be?

The expected value for Grand Total is 63,750,000. This total is achieved by summing the numbers from 1 to 50 and then multiplying by 50,000.

17) Why you are getting different results?

Different results arise due to a race condition with the tot_items variable. As all threads try to modify tot_items simultaneously, the value does not increment as orderly as it should. Consequently, the output varies and does not represent the expected total.