In the first part of Q1, we are asked to use fork() to create two processes. In this case, the memory space is not shared by the processes.
That means that the global variable is different for the two processes. So, the change in the global variable by one process is not reflected in the other.
Hence when the processes are running concurrently, their output is also displayed together and the value of one global variable keeps on decreasing from 10 to -90 whereas
the value of the other keeps on increasing to 100. we used wait() to ensure that the child process is finished before the parent.

In the second part of Q1, we are asked to use pthread_create() to create threads. In this case, since the threads are not independent of each other,
the memory space and code section is shared by the two threads. Here, the change in the global variable by one thread is reflected in the other.
So, if one thread changes the value of the global variable, it is changed for the other thread as well. Hence we ran the threads sequentially but waited for the child thread to end first.

Differences in the output: For the first case, since the variables are independent, the value of the child global variable decreases to -90, and the parent increases to 100.
For the second case, since the variables are dependent, the child process first decreases the value of global variable from 10 to -90, then, the parent process increases the value from -90(since the global variable is the same) to 100.