1. Under what circumstances does a multithreaded solution using multiple kernel threads provide better performance than a single-threaded solution on a single-processor system?
   Ans:

   Well, sometimes page faults may suffer the kernel, and if there is another kernel thread and thus it can switch in to use the interleaving time in a useful manner.

   On the contrary, the single-threaded process will not be able to perform useful task when there is a page fault.

   Therefore, in circumstances where a program might suffer from frequent page faults or has to wait for other system events, a multithreaded solution would have a advantage over a single-processor system.

2. Which of the following components of program state are shared across threads in a multithreaded process?
   A. Register values
   B. Heap memory
   C. Global variables
   D. Stack memory

   Ans:

   By definition, Threads share the **heap, global memory and the page table**. They have private **register values and private stack segments r**espectively.

3. Present a table of runtime that you measured using the time command for running

	Python	C
Without Thread	3m1.380s	0m8.079s
With Thread	5m40.621s	0m6.159s

Surprisingly, the C program outpace Python both with or without Thread. In addition, both programs limit the number of threads to up to 10.

First of all, in C, the threaded program with 6.159s is faster than the unthreaded one with 8.079s. Then, that means that C program take the advantage of the threads in this case.

On the contrary, the python shows the opposite results. And, astoundingly, the unthreaded version is faster than the threaded one. In my opinion, this is because of the Global Interpreter Lock(GIL). Because of the way the interpreter is written, only one thread can safely execute code in the interpreter at the same time. Moreover, the threaded version does not necessarily ensure better performance and from my point of view, it depends on the computing resources and programming languages as well.

In conclusion, the C runs faster than python. The reason C is fast because you have explicit control over allocations and it compiles to native code (among other reasons). The python, however, is easier to write code and development. So, it is a trade-off between the performance and development.