Problem 2:

**- In which cases we should use aligned\_malloc() instead of standard malloc.**

Standard malloc() or realloc() in GNU systems always return address which is a multiple of eight (or sixteen on 64-bit systems), mean that malloc will return a suitably aligned memory block for any of the standard types.

But if you need a stricter alignment (for example, on SSE2 (SIMD) instructions need their data aligned on 16-byte boundaries, or a memory alignment is needed for better performance on some system), you’ll need to use aligned\_malloc()

**- How can we increase the size of heap in a running process?**

Heap in a running program is a continuous (in term of virtual addresses) space of memory with three bounds: a starting point, a maximum limit and an end point called break. You can further increase the size of heap by moving the end point farther with sbrk(), or place the break at a farther address using brk(). But, a user program shouldn’t, and won’t need to call these system calls, as these system calls will be called when using malloc().

However, we can write a simple program to try increase the heap size by using sbrk()

