PROBLEM 2

First of all, malloc(size) in C programming language is used to allocate a block size bytes of memory, and size is arbitrary. It is not optimal if we allocate block with a size that is not fixed in pages and it also waste a lot of space in obsolete memory chunks. In other words, aligned_malloc will create more memory fragmentation, in that way, we can optimize the time that the program requires. About pages, if we do not care about alignment, we maybe create some data block that lies on 2 pages, then both only can free() until this data exists.

While running, the process can increase the size of the heap by syscall sbrk(intptr_t incr). This function can move the break of the heap by the amount of space that called in function. E.g.:

```
void* brk;


   brk = sbrk( 0x3100 );
   printf( "New break value after sbrk( 0x3100 ) \t%p\n",
           brk );

   brk = sbrk( 0x0200 );
   printf( "New break value after sbrk( 0x0200 ) \t%p\n",
           brk );
   return 0;
```

First, we expand the size of the heap more than 0x3100 and print the last break. Then we expand the heap 0x0200 and print the break that is changed by first sbrk() function.

```
OS/Lab04 git/master*
❯ ./test
New break value after sbrk( 0x3100 )    0x563baa7ad000
New break value after sbrk( 0x0200 )    0x563baa7b0100
```

This code proved that we can change the break of heap. It means that code increased the size of heap.