malloc_f(long size)

size = size + sizeof(MCB);
sizeof( ) is a macro function whose return value is calculated at compile time. As you look at class MCB that includes only two integers, each with 4 bytes, sizeof(MCB) will be replaced with 8 by compiler.

```
// scan from the top of the heap
for ( void * cur = heap_top; cur < heap_end; cur = cur + cur_mcb->size ) {
}
```

This “cur = cur + cur_mcb->size” statement is a generic idea to advance to the next mcb.

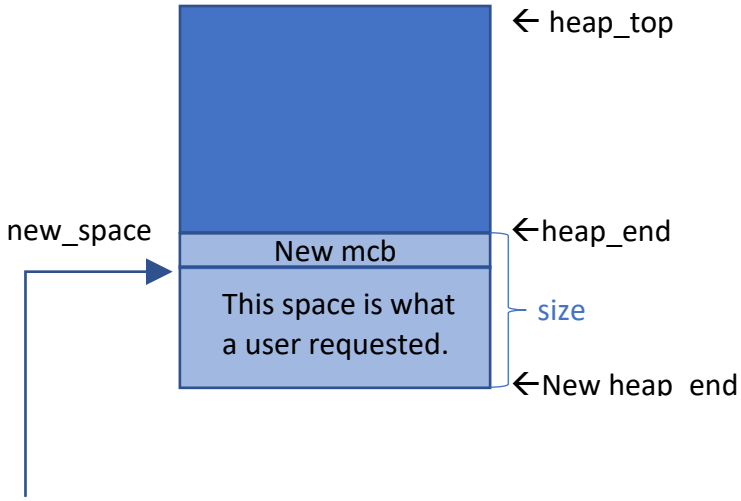
However, it will cause a compilation error. How can you add cur_mcb->size to cur?
You need a correct type casting. Since most computers are now based on 64-bit addressing, a pointer (e.g., void *cur) is a 64-bit variable. So, for an arithmetic operation, you need to cast cur to (unsigned long long int).

For each cur_mcb, you got to check cur_mcb->available and cur_mcb->size fits size. If so, you have to:
Disable cur_mcb->available.
Let new_space point to cur or cur_mcb. (Probably, letting it point to cur is easy as both variables are void*.)

```
If ( new_space == NULL ) {
}
```

This means that you couldn’t find a good mcb that satisfies a user request. You need to ask OS to extend head_end with sbrk(size), because you need a new mcb and a space the user requested. Recall size = size + sizeof(MCB)!

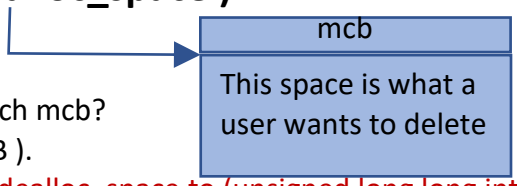
Don’t update heap_end immediately. This is because new_space should point to heap_end. Thereafter heap_end should be moved down.



Finally, malloc_f( ) will return as follows

```
    // new space is after new MCB
      return (void *)( ( long long int )new_space + sizeof( MCB ) ); // actually unsigned long long int is safter!
```

free_(void *dealloc_space)



So, how can you reach mcb?
Subtract sizeof(MCB).
Don’t forget to cast dealloc_space to (unsigned long long int).