Void *a;

int*p;

p = a ✓

a = p ✗

a = (void*)p

Void *a = 1000;

int *p = 2000;

p+1 -> 2004

a+1 -> 2001

(int)a = 12; ✓

*a = 12 ✗

int *p = malloc(100);

Assignment 5 : Implementing memory allocation

```
void* mem_alloc(int size) {
    // 1 -> align to 4 bytes
    // 2 -> find best-fit block
    // 3 -> split the block
    // 4 -> return pointer to payload

    size = size aligned to 4-bytes

    // find the best-fit
    block *best_block = list_head;
    for each block B in block_list {
        if(B is not allocated and
           B is bigger than size and
           B is smallest so far
                (smaller best_block) )

            best_block = B;
    }
```

if no best block found:

return error → (out of memory)

// split the best block

block* B = best_block

~~if (B.size > ??)~~

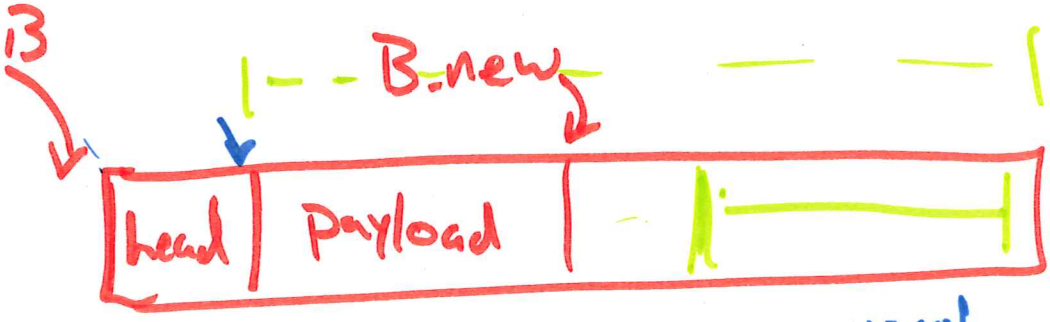
if enough space in B for another block {

↳ if (B.size - size > ??)

↓ size w/ header + sizeof(block_header)

↳ 4 (min payload)

block* B_new = (void*) B + sizeof(header) + size



B_new.size = B.size - header - size

~~sizeof~~

B_new.next = B.next

B.next = B_new

B.allocated = true

}

return (void*) B + sizeof(header)

```
int free (void * ptr)
    //check for errors
    if (ptr == NULL) return error

    block_header * B = ptr - sizeof (header)
    if (not B.allocated) return error

    Set B.allocated to false

    //coalesce free blocks
    for each block C in block list {
        if C.free and C.next.free
            coalesce C and C.next
        else
            go to next C
    }

    return no error
```

if you coalesce C + C.next. Re-run the loop on new C