TD6 : MPC

```
char buffer[BUFFERSIZE]
```

```
struct block_m {
    block_t * prev;
    block_t *next;
    int magic_number;
    unsigned int blocksize;
    void* userspace; <- très important, dernier element.
}
typedef struct block_m block_t;
static block_t *freelist;
static block_t *allocatedlist = NULL;
```

```
static void init() {
    freelist = (block_t *) buffer;
    freelist -> prev = freelist -> next = NULL;
    freelist -> magicnumber = MAGIC;
    freelist -> userspace=(void *) ((char *)freelist
                                    + sizeof(block_t));
    freelist -> blocksize = BUFSIZE - sizeof(block_t);
    return;
}

#define MAGIC 0xAE2795
#define MARGE 2* sizeof(int)
```

```
block_t *extraire(block_t **l, unsigned int size) {
    block_t *p;
    if (*l == NULL) return NULL;
    if ((*l) -> blocksize < size) return NULL;
    if ((*l) -> blocksize < size + sizeof(block_t) + MARGE) {
        p = (*l)
        if ((*l) -> prev == NULL) {
            freelist = p -> next;
            (p -> next) -> prev = NULL;
        } else {
            (p -> next) -> prev = p -> prev;
            (p -> prev) -> next = p -> next;
        }
        p -> prev = p -> next = NULL;
    }
    Return p;
}
```

```
    p = (block *)((char *)*l + (*l)->block_size - size);

    (*l)->block_size = (*l)->block_size - ?
    p->magic_number -> MAGIC;
    p->prev = p->next = NULL;

    trier (&(free_list));
    Return p;
  }
  extraire (&((*l)->next)), size);
}

void inserer(block_t **l, block_t *b) {
    if (*l == NULL) {
        *l = b; Return;

    if ((*l)->next == NULL)
        (*l)->next = b; Ret

    if ((*l)->block_size < b->block_size) {
        (*l)->prev = b;
        b->next = *l;
        *l = b; Ret
    }
    inserer (&((*l)->next), b);
}

void * monmalloc (unsigned int size) {
    block_t* monbloc;

    monbloc = extraire (&free_list, size);
    if (monbloc == NULL) Return NULL;
    inserer (&allocated_list, monbloc);
    return monbloc -> userspace;
}
```

```
void free(void *ptr){
    block_t *p;
    p=(block_t *)((char *)ptr-sizeof(block_t));
    if(p->magicnumber != MAGIC NUMBER)
        return;
    p->next = p->prev = NULL;
    extraire(&busylist, p);
    inserer(&freelist, p);
    return;
}
```