

```

1 #include "malloc.h"
2 #include <sys/mman.h>
3 #include <stdio.h>
4 #include <stdint.h>
5
6 uint64_t* HEAP_START = NULL;
7 #define HEAP_SIZE_BYTES 240
8 #define HEAP_WORDS HEAP_SIZE_BYTES/8
9
10 void init_heap() {
11     HEAP_START = mmap(NULL, HEAP_SIZE_BYTES, PROT_READ | PROT_WRITE,
12         MAP_PRIVATE | MAP_ANONYMOUS, -1, 0);
13     HEAP_START[0] = HEAP_SIZE_BYTES;
14 }
15
16 void print_heap() {
17     int i = 0;
18     uint64_t* start = HEAP_START;
19     while(i < HEAP_WORDS) {
20         uint64_t h = start[i];
21         if(h % 2 == 0)
22             printf("%p : %llu [%llu bytes FREE]\n", &start[i], h, h);
23         else
24             printf("%p : %llu [%llu bytes BUSY]\n", &start[i], h, h - 1);
25         i += h / 8;
26     }
27 }
28
29 // Returns a pointer to the header of a block of at least size bytes
30 uint64_t* find_block(uint64_t* start, size_t size);
31
32 /* Splits the block into two, one busy, one free.
33    Note also that the current block size and size are 8-byte aligned */
34 void split_block(uint64_t* block_start, size_t size);
35
36 /* Returns the size of a block for a given malloc request:
37    rounded up to nearest 8, plus 8 for header */
38 size_t block_size_of(size_t user_request);
39
40 void* malloc(size_t size) {
41     if(HEAP_START == NULL) { init_heap(); }
42     size_t block_size = block_size_of(size);
43     uint64_t* block_to_use = find_block(HEAP_START, block_size);
44     if(block_to_use == NULL) {
45         printf("Could not allocate %ld\n", block_size);
46         return NULL;
47     }
48     split_block(block_to_use, block_size);
49     void* addr_for_user = &block_to_use[1];
50     return addr_for_user;
51 }
52
53 void free(void* ptr) {
54     uint64_t* p = ptr;
55     p[-1] = p[-1] & (~1);
56 }

```

```

1 #include "malloc.h"
2 #include <stdio.h>
3
4 int main() {
5     char* a = malloc(10);
6     char* b = malloc(37);
7     int* c = malloc(7 * sizeof(int));
8     printf("After allocating a, b, c:\n");
9     print_heap();
10    free(b);
11    int* d = malloc(5 * sizeof(int));
12    printf("\nAfter freeing b and allocating d:\n");
13    print_heap();
14 }

```

```

$ gcc -g malloc.c test.c -o test
$ ./test
After allocating a, b, c:
0x1043d4000 : 25 [24 bytes BUSY]
0x1043d4018 : 49 [48 bytes BUSY]
0x1043d4048 : 41 [40 bytes BUSY]
0x1043d4070 : 128 [128 bytes FREE]

After freeing b and allocating d:
0x1043d4000 :

0x1043d4018 :

0x1043d4038 :

0x1043d4048 :

0x1043d4070 :

```

```
size_t block_size_of(size_t user_request) {

}

uint64_t* find_block(uint64_t* start, size_t size) {

}

void split_block(uint64_t* block_start, size_t size) {

}
```