```c
#include <stdlib.h>
#include <stdio.h>
#include <stdint.h>
#include <sys/mman.h>

#define HEAP_SIZE 400
#define VAL_SIZE 8

uint64_t* HEAP_START = NULL;

void init_heap() {
...
}

void* my_malloc(size_t size) {
  uint64_t* current = HEAP_START;
  while(current < (HEAP_START + (HEAP_SIZE / VAL_SIZE))) {
    uint64_t cur_header = *current;
    uint64_t cur_size = (cur_header / 2) * 2;
    // want to know (a) size (b) is it free
    if((cur_header % 2 == 0) && (size <= cur_size)) { // check if free
      // GOAL: split up the block into the malloc'd part and the free part
      // Round up size to next multiple of 8
      size_t rounded = ((size + 7) / 8) * 8;
      *current = rounded + 1; // rounds up and sets the "busy" bit

      // what if remaining is close to 0/8/16, etc
      size_t remaining = cur_size - (rounded + VAL_SIZE);
      uint64_t* remaining_ptr = current + (rounded / VAL_SIZE) + 1;
      *remaining_ptr = remaining;

      return current + 1;
    }
    else {
      uint64_t* next = current + (cur_size / VAL_SIZE) + 1;
      current = next;
    }
  }
  return NULL;
}

void print_heap() {
  uint64_t* current = HEAP_START;
  while(current < (HEAP_START + (HEAP_SIZE / VAL_SIZE))) {
    uint64_t cur_header = *current;
    uint64_t cur_size = (cur_header / 2) * 2;
    printf("%p\t%d\t%d\n", current, cur_header % 2, cur_size);
    uint64_t* next = current + (cur_size / VAL_SIZE) + 1;
    current = next;
  }
  printf("\n\n");
}

void my_free(void* p) {
  uint64_t* current = p;
  uint64_t* header = current - 1;
  if(*header % 2 == 1) { // else case: valgrind reporting double free!
    *header = *header - 1;
  }
}

int main() {
  init_heap();
  int* a = my_malloc(40);
  int* b = my_malloc(10);
  int* c = my_malloc(20);
  my_free(b);
  print_heap();
  int* d = my_malloc(30);
  print_heap();
  int* e = my_malloc(12);
  print_heap();
}
```

```
$ gcc -g mem.c -o mem
$ ./mem > out.txt
^C
$ head -n 20 out.txt
0x7f8b17aed000  1       40
0x7f8b17aed030  0       16
0x7f8b17aed048  1       24
0x7f8b17aed068  0       288


0x7f8b17aed000  1       40
0x7f8b17aed030  0       16
0x7f8b17aed048  1       24
0x7f8b17aed068  1       32
0x7f8b17aed090  0       248


0x7f8b17aed000  1       40
0x7f8b17aed030  1       16
0x7f8b17aed048  0       -8
0x7f8b17aed048  0       -8
0x7f8b17aed048  0       -8
0x7f8b17aed048  0       -8
0x7f8b17aed048  0       -8
```