```
CS 341 Lecture #9
Building an allocator
```

1. The following allocator will use this linked list structure:

```
typedef struct _metadata_entry_t {
    void *ptr;
    int size;
    int free; //o(in use) or 1(available)
    struct _metadata_entry_t *next;
    } metadata_entry_t;
```

Global variable:

```
o7 static metadata_entry_t * head = NULL;
```

2. Complete malloc()

```
void *malloc(size t size) {
08
09
10
          /* See if we have free space of enough size. */
          metadata entry t *p = head;
11
          metadata entry t *chosen = NULL;
12
13
          while (p != NULL) {
14
            if (p->free && _____) {
  if (chosen == NULL || (chosen && p->size < chosen->size)) {
15
16
                 chosen = p;
17
18
19
20
            p = p - next;
21
22
          if (chosen) {
23
            chosen->free = o;
24
            return chosen->ptr;
25
26
27
         /* Add our entry to the metadata */
28
          chosen = sbrk(o):
29
          sbrk(sizeof(metadata_entry_t));
30
          chosen->ptr = sbrk(o);
31
          if (sbrk(size) == (void*)-1) {
32
            return NULL;
33
34
          chosen->size = size:
35
          chosen->free = o:
36
37
          chosen->next = head;
38
          head = chosen;
39
40
          return chosen->ptr;
41
```

3. Complete free()

```
01
       void free(void *ptr) {
02
        if (!ptr) return;
03
04
         metadata entry t *p =
05
         while (p) {
06
          if (p->ptr == ptr) {
07
08
09
          p = p->next;
10
11
12
13
         return;
14
```

Which placement algorithm does this malloc() use?

Is calling sbrk 4 tims necessary?

What is the order of growth running time for this implementation of free?

- 4 i) Why does this implementation suffer from false fragmentation?
- ii) When should we split blocks?
- iii) Does this implementation use an explicit or implicit linked list?

5. How would you change malloc() to use a first-fit placement allocation?

8. Towards a better allocator

Implementing realloc & improving performance of free()

Hint: Can we ensure this structure is immediately before the user's pointer?

```
typedef struct _metadata_entry_t {
void *ptr;
int size;
int free;
struct _metadata_entry_t *next;
end
metadata_entry_t;
```

We want an O(1) deallocator!

End of the allocator challenge?

- 1. Block Spitting & Block Coalescing
- 2. Memory pools
- 3. Advanced: Slab allocator and Buddy allocator
- 4. Internal vs External Fragmentation
- 5. How we use Boundary Tags to implement coalescing?

9. Puzzle:

Complete this code to read in values from stdin into heap memory. Can you beat CS225 code by using C and realloc to increase the size of the array? Fix any errors you notice.

```
#define quit(mesg) {puts(mesg); exit(1);}
02
    size_t capacity = 256;
    size t count = 0;
os int* data = malloc( capacity );
    if(! data) quit("Out of memory");
07
    while(!feof(stdin) && !ferror(stdin)) {
08
      if( count == capacity) {
        capacity *= 2;
10
11
12
      if(fscanf(stdin, "%d", data+count)!= 1) break;
13
14
      count++;
15
    // can now reduce capacity to the number actually read
16
    printf("%d values read",(int) count);
    data = realloc(data, count);
```