```c
typedef struct _pool_t {
    size_t capacity;
    size_t used;
    char buff[]; // C99. Must be last
} pool_t;

void* create_pool(size_t capacity) {
// fix the mistake(s)
    pool_t * result = malloc(capacity + _____);
    assert(result);
    result -> capacity = capacity;

    memset( result _____, 0x5a, capacity);

    return result;
}

void* allocate(pool_t* pool, size_t request) {
  assert(pool);
// How would you round to ensure request is a multiple of sizeof(size_t) ?

   request _____

// Leave space for our meta data...
   char * result = pool->buff + used + sizeof(size_t) * 2;

   // Todo: Round up to ensure natural alignment e.g. result%16 is 0.


   result _____

   pool->used = request + (result - pool->buff); // Is this correct?

   assert( *result == _____ );
   size_t* bounds = (size_t*) result;
   bounds[-2] = request;
   bounds[-1] = 0xdeadbeef'; // Why this way round?

   bounds[ request ] = 0xBAADF00D; // Fix the error

   return result;
}

void deallocate(pool_t* pool, void* ptr) {
  assert(pool && ptr);
  size_t *bounds = ptr;

  assert(bounds[-1] == _____);
  size_t size = bounds[-1];

  assert( _____);
  memset(ptr, 0x5a, size);
}
```

```
Advanced techniques
```

0. Advantages of memory pools?

1. Additional explicit linked list: Store memory addresses of next free link

- Store free blocks pointers inside the unused space of the free block. More work to do during free()
- Free Block list can now be in arbitrary searchable order.
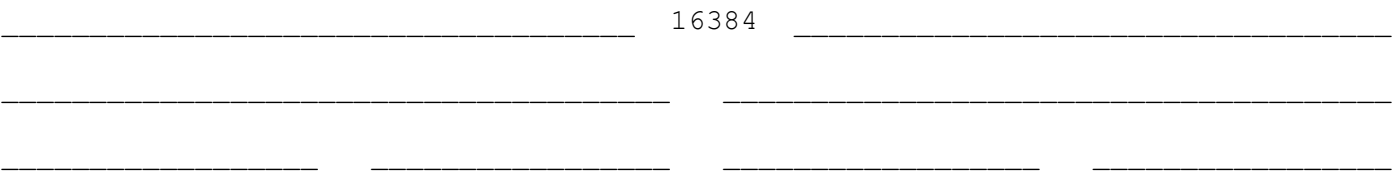- Still use boundary tags to implement coalescing

| 32 | _____ | 32 | 24 | _____ | 24 | 96 | _____ | 96 |

2. Segregated free list: Different lists for different sizes. Advantage?

3. Where would you find a slab allocator?

4. Advantages of deferred coalescing?

5. Buddy Allocator & Fragmentation

_____   16384   _____

_____   _____

_____   _____   _____   _____

```
malloc(3000)
```