CS107, Lecture 24
Managing The Heap, Wrap

Reading: B&O 9.9 and 9.11

Ed Discussion: https://edstem.org/us/courses/46162/discussion/3875438

Final Assignment: Explicit Allocator

- **Must have** headers that track block information like in implicit (size, status in-use or free) – you can copy from your implicit version
- **Must have** an explicit free list managed as a doubly-linked list, using the first 16 bytes of each free block's payload for next/prev pointers.
- **Must have** a malloc implementation that searches the explicit list of free blocks.
- **Must** coalesce a free block in free() whenever possible with its immediate right neighbor.
- **Must** do in-place realloc when possible. Even if an in-place realloc is not possible, you should still absorb adjacent right free blocks as much as possible until you either can realloc in place or can no longer absorb and must realloc elsewhere.

Final Project Tips

Read B&O textbook.

- Offers some starting tips for implementing your heap allocators.
- Make sure to cite any design ideas you discover.

Honor Code/collaboration

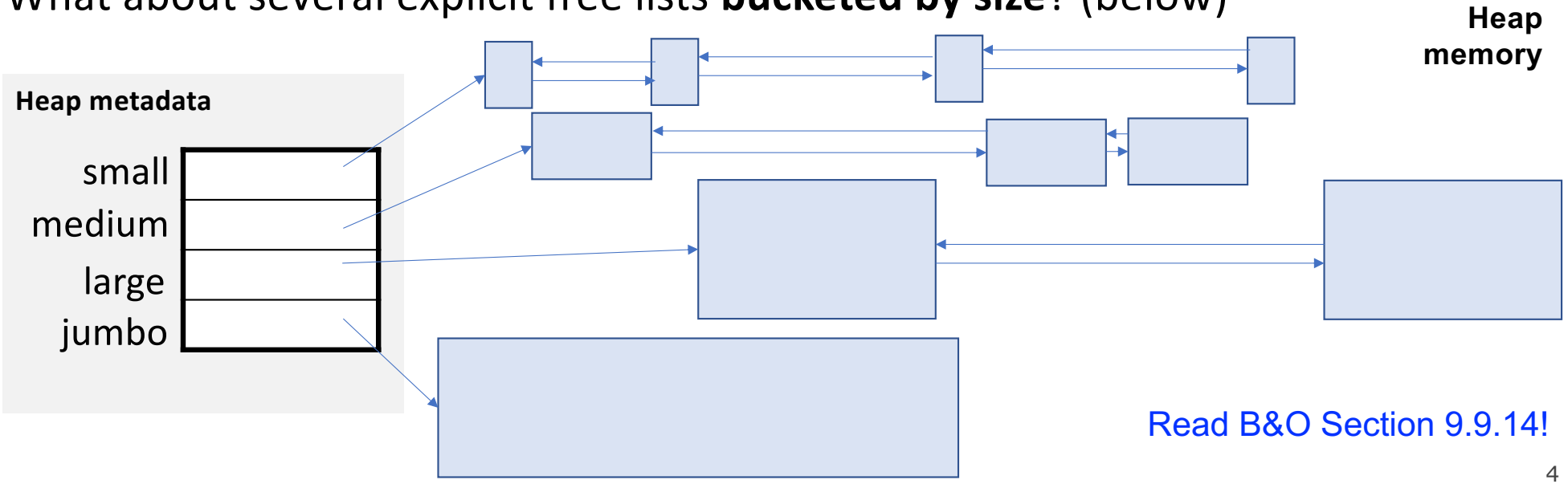
- All non-textbook code is off-limits.
- Please do not discuss discuss code-level specifics with others.
- Your code should be designed, written, and debugged by you independently.

Helper Hours

- We will provide good debugging techniques and strategies!
- Come and discuss design tradeoffs!

Going beyond: Explicit list w/size buckets

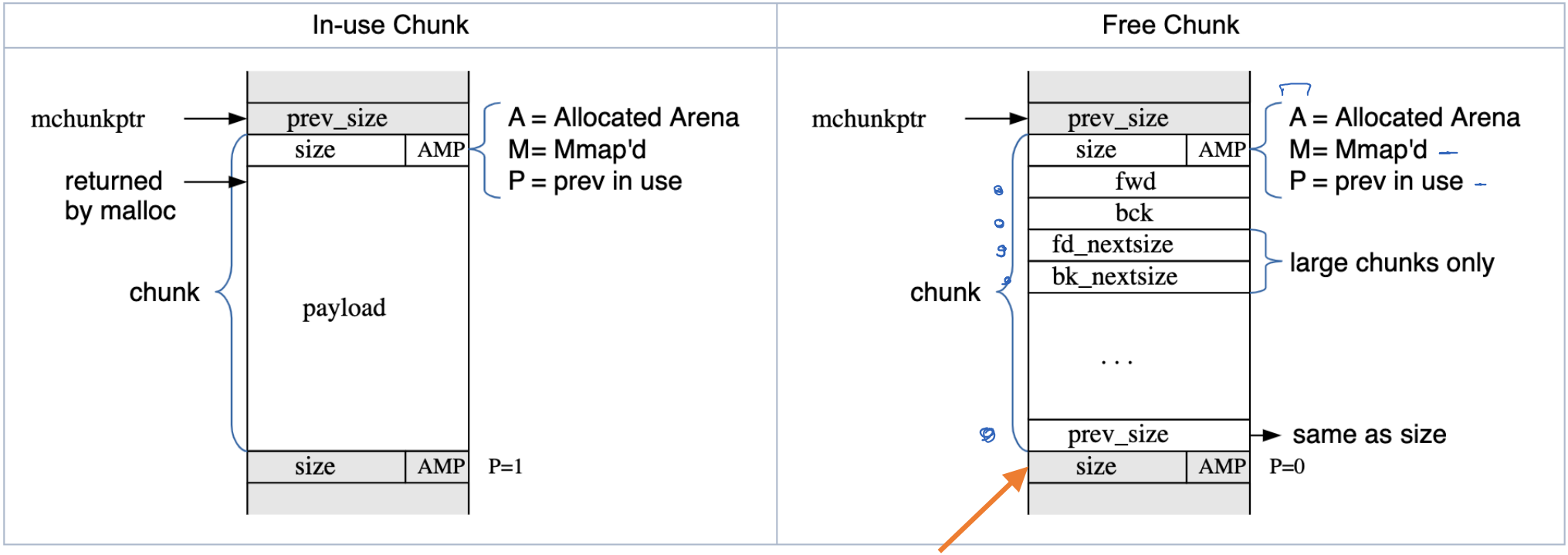
- Explicit lists are much faster than implicit lists.
- However, a first-fit placement policy is still linear in total # of free blocks.
- What about an explicit free list **sorted by size** (e.g., as a tree)?
- What about several explicit free lists **bucketed by size**? (below)



Read B&O Section 9.9.14!

In the wild: glibc allocator

- https://sourceware.org/glibc/wiki/MallocInternals

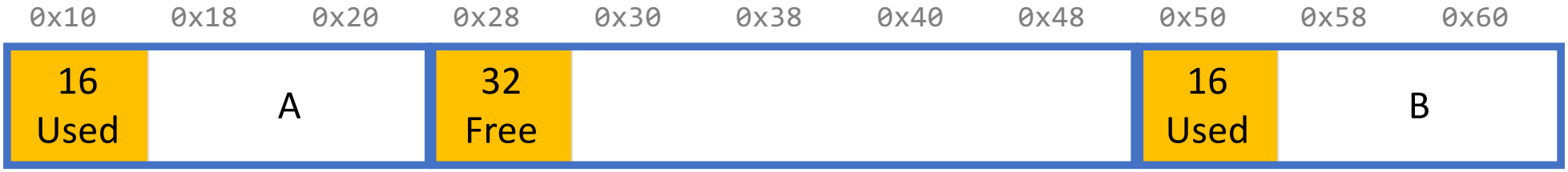


Footer/Boundary tag (see textbook)

Practice 1: Explicit (realloc)

optional but extra practice

For the following heap layout, what would the heap look like after the following request is made, assuming we are using an **explicit** free list allocator with a **first-fit** approach and **coalesce on free + realloc in-place**?

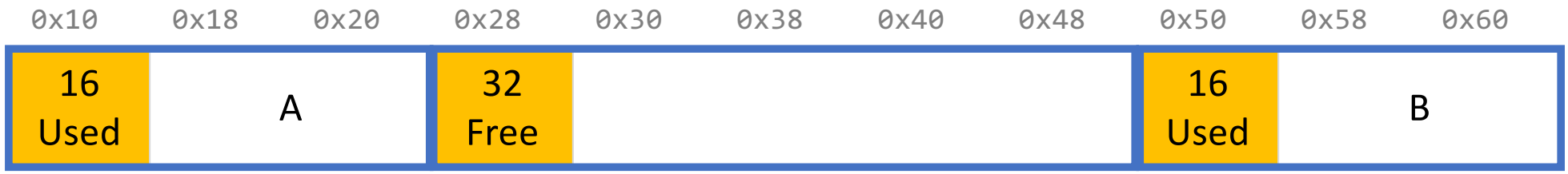



```
realloc(A, 24);
```

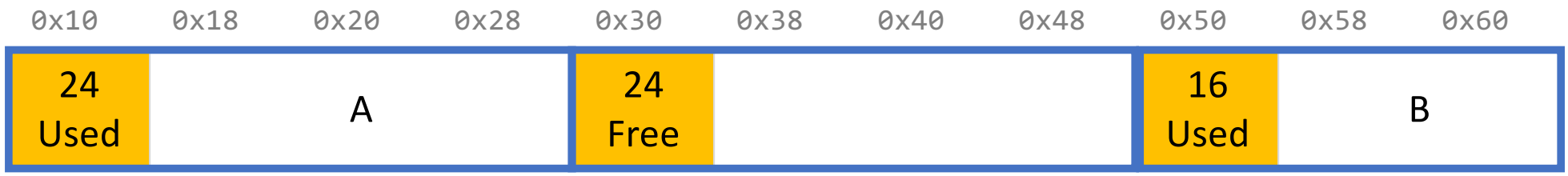
Practice 1: Explicit (realloc)

optional but extra practice

For the following heap layout, what would the heap look like after the following request is made, assuming we are using an **explicit** free list allocator with a **first-fit** approach and **coalesce on free + realloc in-place**?



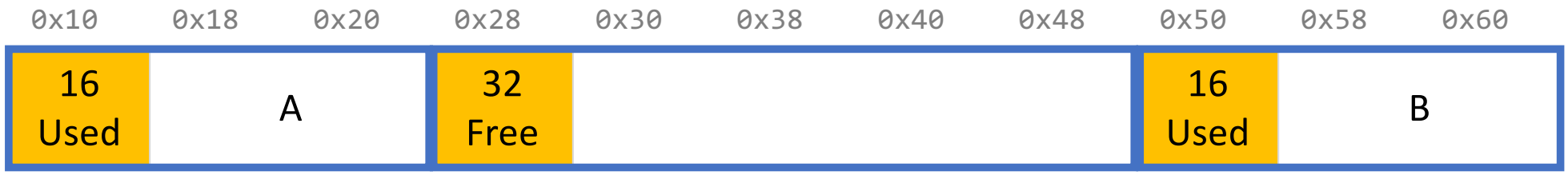

```
realloc(A, 24);
```



Practice 2: Explicit (realloc)

optional but extra practice

For the following heap layout, what would the heap look like after the following request is made, assuming we are using an **explicit** free list allocator with a **first-fit** approach and **coalesce on free + realloc in-place**?

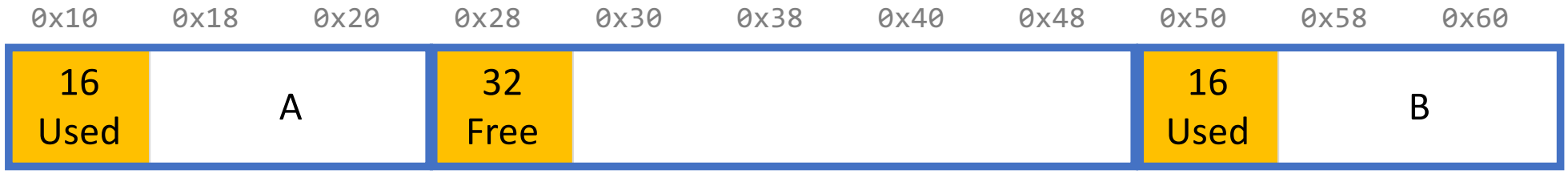



```
realloc(A, 56);
```

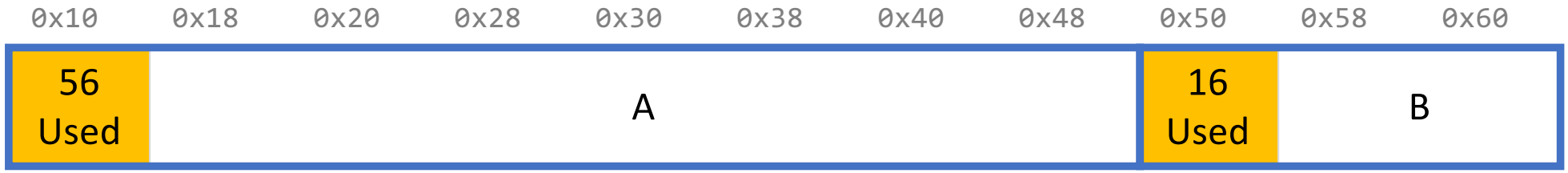
Practice 2: Explicit (realloc)

optional but extra practice

For the following heap layout, what would the heap look like after the following request is made, assuming we are using an **explicit** free list allocator with a **first-fit** approach and **coalesce on free + realloc in-place**?



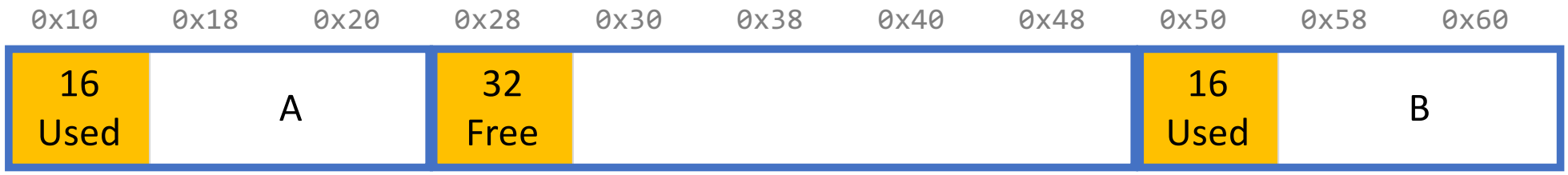

```
realloc(A, 56);
```



Practice 3: Explicit (realloc)

optional but extra practice

For the following heap layout, what would the heap look like after the following request is made, assuming we are using an **explicit** free list allocator with a **first-fit** approach and **coalesce on free + realloc in-place**?

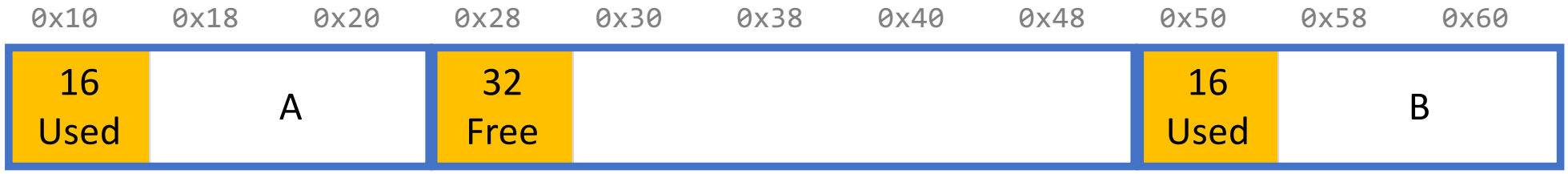



```
realloc(A, 48);
```

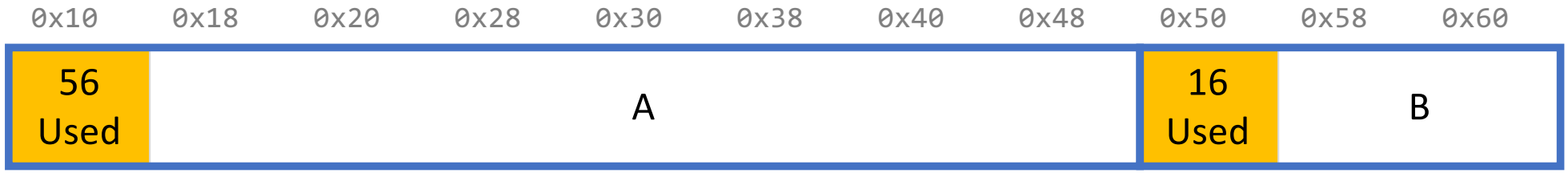
Practice 3: Explicit (realloc)

optional but extra practice

For the following heap layout, what would the heap look like after the following request is made, assuming we are using an **explicit** free list allocator with a **first-fit** approach and **coalesce on free + realloc in-place**?



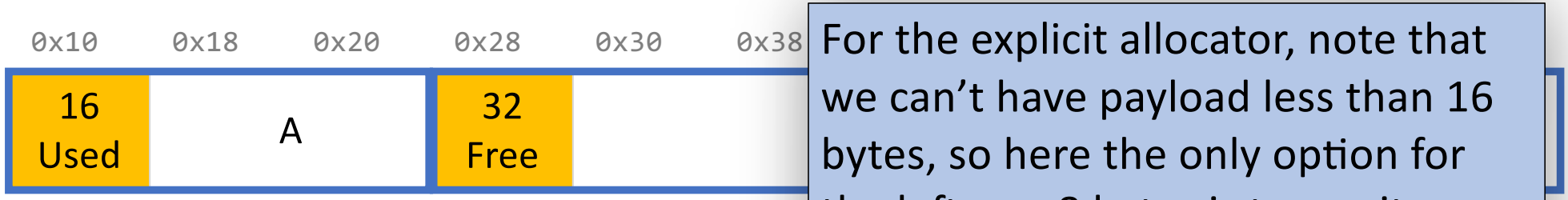

```
realloc(A, 48);
```



Practice 3: Explicit (realloc)

optional but extra practice

For the following heap layout, what would the heap look like after the following request is made, assuming we are using an **explicit** free list allocator with a **first-fit** approach and **coalesce on free + realloc in-place**?



For the explicit allocator, note that we can't have payload less than 16 bytes, so here the only option for the leftover 8 bytes is to use it as padding for the existing block.

```
realloc(A, 48);
```