Implicit List: Finding a Free Block

■ *First fit:*

- Search list from beginning, choose *first* free block that fits:

```
p = start;
while ((p < end) &&      \\ not passed end
       ((*p &.1) ||      \\ already allocated
       (*p <= len)));    \\ too small
  p = p + (*p                          ock (word addressed)
```

- Can take linear tim   ted and free)
- In practice it can cause "splinters" at be

■ *Next fit:*

- Like first fit, but search list starting where previous search finished
- Should often be faster than first fit: avoids re-scanning unhelpful blocks
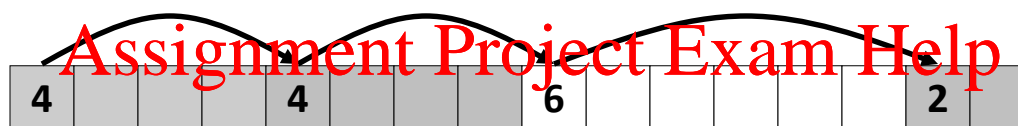- Some research suggests that fragmentation is worse

■ *Best fit:*

- Search the list, choose the *best* free block: fits, with fewest bytes left over
- Keeps fragments small—usually helps fragmentation
- Will typically run slower than first fit
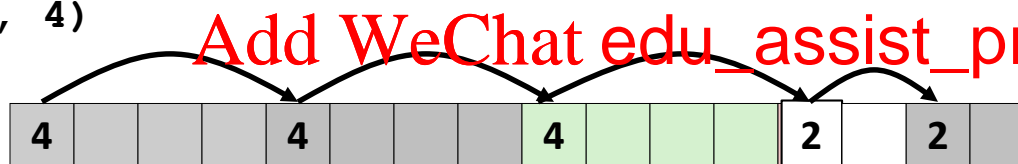
Implicit List: Allocating in Free Block

■ **Allocating in a free block: *splitting***

■ Since allocated space might be smaller than free space, we might want to split the block

Assignment Project Exam Help

https://eduassistpro.github.io/

`addblock(p, 4)`

Add WeChat edu_assist_pro

```
void addblock(ptr p, int len) {
  int newsize = len;
  int oldsize = *p & -2;                // mask out low bit
  *p = newsize | 1;                     // set new length
  if (newsize < oldsize)
    *(p+newsize) = oldsize - newsize;   // set length in remaining
}                                       //   part of block
```