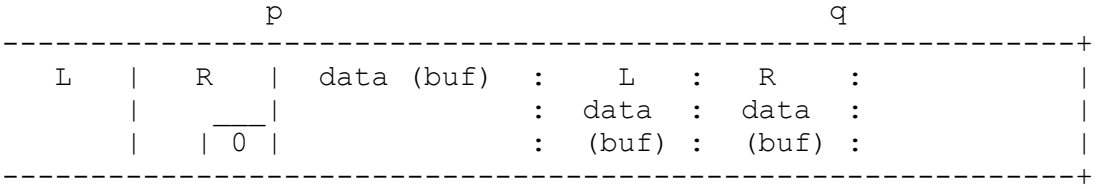
We left off section with the following diagram at point C, where we've allocated p to be the length of both blocks, and obsd_strlcpy'd our attack string to the data.

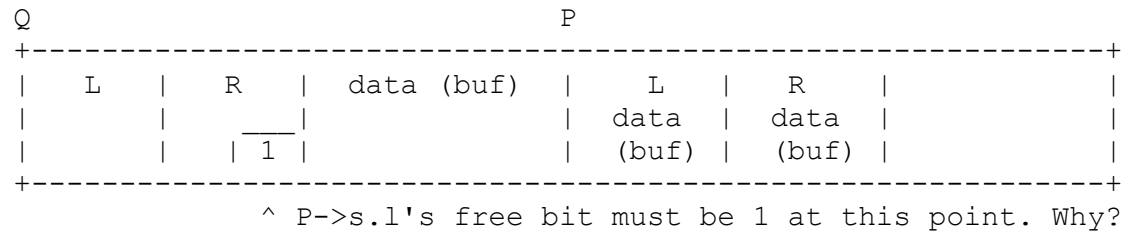


Now, when we execute the code in question 6 in the double free of q, we're trying to coalesce leftward.

We call in target5.c: tfree(q)
But in tmalloc.c, the function signature is tfree(vp). Be careful as the "q" variable from the target is now "vp". I'll be changing the names to Q (for q's left chunk tag) and P (for q's chunk tag).

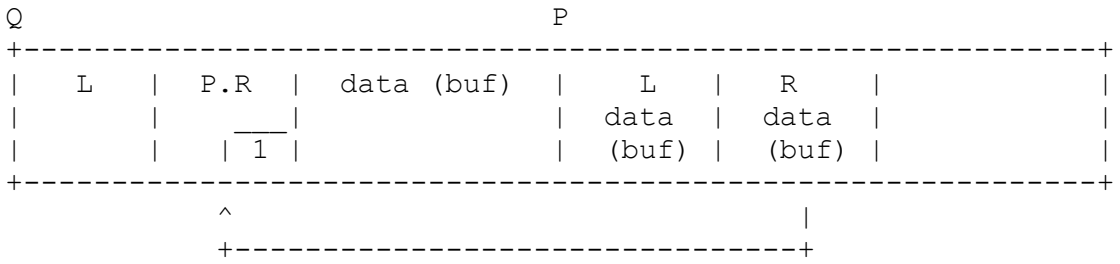
```
99    void tfree(void *vp)
...
106   p = TOCHUNK(vp);
...
108   q = p->s.l;          <- This value is affected by obsd_strlcpy.
...
112   q->s.r = p->s.r;     <- These values are also affected by obsd_strlcpy.
113   p->s.r->s.l = q;
```

At line 112, we have the following diagram:



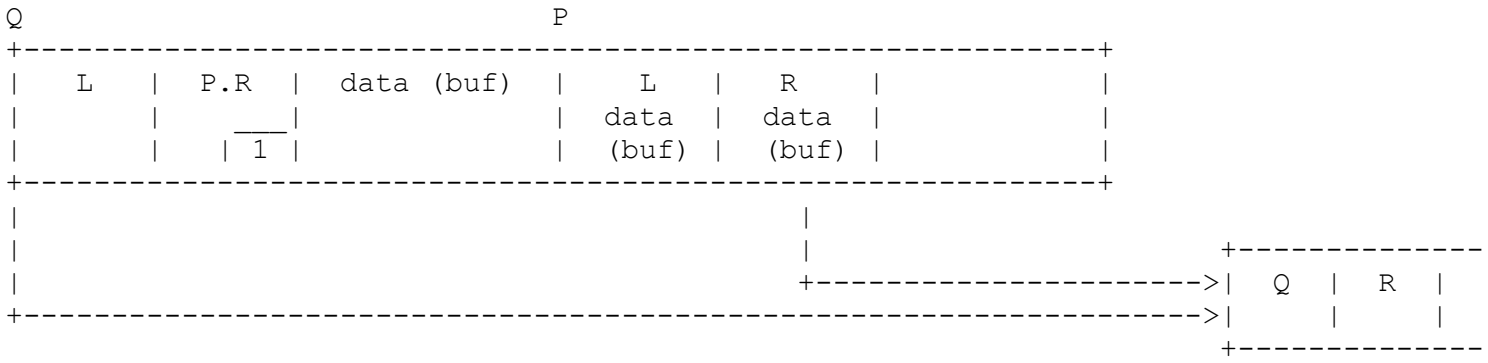
Line 112 will copy the value of P's right pointer to the value stored by Q's right pointer.

Result:



Line 113 will copy the value of Q to the value stored by P's right pointer's left pointer. P’s right pointer is interpreted as a valid heap memory chunk with its own left and right pointer.

Result:



Since the q that's double freed lies within the attack string we specify, we own the value of q's left and right pointers. So, we can write the value of Q's left pointer (address of buffer?) to a location we specify (location of saved EIP?).