CSE 484 In-section Worksheet #3

Q1. Which `gdb` command allows us to:

view the four words starting at `ebp` in hex? x/4wx $ebp

view the next five instructions at `eip`? x/5i $eip

view all instructions for function `foo`? disas foo

Q2. What happens to the stack when the x86 instruction `RET` is called?

"The ret instruction implements a subroutine return mechanism. The instruction first pops a code location off the hardware supported in-memory stack. It then performs an unconditional jump to the retrieved code location" (https://www.cs.virginia.edu/~evans/cs216/guides/x86.html).

TL;DR: The stack pops off the next address and resumes execution there (presumably, this is the instruction right after the function call that just exited).

Q3. What do `tmalloc()` and `tfree()` do?

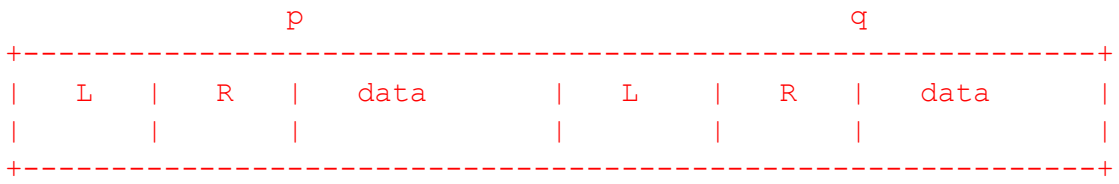
See slide 9.

Q4. What's the issue with this code?

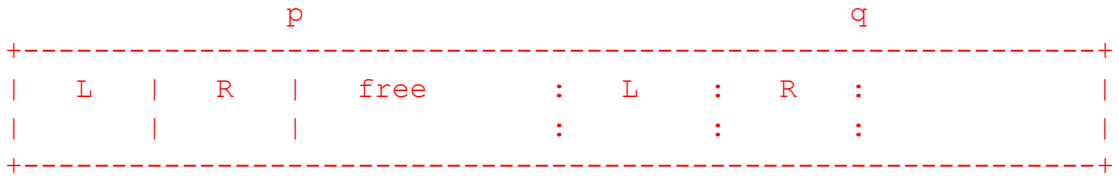
```
      char *p; char *q;
      if ( (p = tmalloc(128)) == NULL)
      { exit(EXIT_FAILURE); }
      if ( (q = tmalloc(128)) == NULL)
      {exit(EXIT_FAILURE); }
A
      tfree(p);
      tfree(q);
B
      if ( (p = tmalloc(256)) == NULL)
      {exit(EXIT_FAILURE); }
      obsd_strlcpy(p, arg, 256);
C
      tfree(q); Double free!
```

Q5. Based on `tmalloc.c`, draw what the heap/free list looks like at points, A, B, and C. Include chunk structure and label `p` (at or before point B), `p` (at point C), and `q`. Where is `buf` copied?

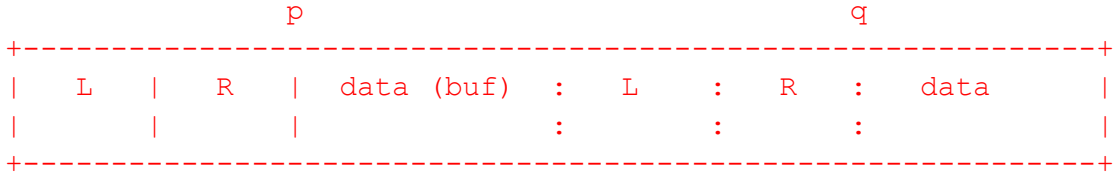
A:
p, q = tmalloc(128) (two blocks allocated.)



B:
tfree(p), tfree(q) (q's L and R are implicit because the two freed block pointers have been coalesced into one block)



C:
obsd_strlcpy(p, arg, 256) (copies over data, including overwriting the block pointers of "q")



Q6. Given your diagrams and the following code for chunk consolidation (from `tmalloc.c`), what do the following statements do when executed in the call `tfree(q)` after point C?

```
q->s.r = p->s.r;
p->s.r->s.l = q;
```

The above is from lines 112 and 113 in tmalloc.c.

If we control chunks p (and q), this code will write the value of q (address of buffer?) to a location we specify (location of saved EIP?).

Be careful about variable names p and q when working with tfree- the q you double free is the p in tfree. The q in this snippet is not the same as p in the diagram- q here is set to be p (aka q in the target code)'s right pointer in line 106. In the first line, the left block's right pointer is being set to the right pointer of the right block (as it should when coalescing leftward).