1. a.

Use after free:

char \*c = malloc(10);

free(c);

c[0] = 'a';

Buffer overflow:

char \*c = malloc(10);

c[10] = 'a';

Null dereference:

struct t { int a; }

struct t \*t\_ptr = null;

t\_ptr->a = 0;

1. b.

Since b is replaced by a, and a is always 0 after calculating a = a - b, the compiler can rewrite foo as:

int foo(int input) {

int c;

if (input >= 0)

c = 0;

else

c = input / 0;

return c;

}

Dividing by 0 is undefined behavior so the compiler can infer that the else branch is never taken, and simplify the function to just return 0. It may also then inline the result of the function and then since += 0 is a useless operation it may also completely optimize away the loop and just print count as 0.

Since the program contains undefined behavior, the compiler is technically allowed to do whatever it wants, and so the crash observed at -O3 can't be a mis-compilation.

1. c. not covered

1. d. not covered

2. not covered

3.