Objectives for 10/2/2017 (Mon)

- Exam 1 will be closed book – vote was 81 closed to 7 open
- Memory faults
  - leaks (“definitely lost”)
  - buffer overflow (“invalid write”)
  - buffer overread (“invalid read”)
  - Valgrind messages:
    - “HEAP SUMMARY: in use at exit: □ bytes in □ blocks”
    - “total heap usage: □ allocs, □ frees, □ bytes allocated”
- Allocation types
  - `TYPE NAME = ...` (inside a function, with no qualifiers) → allocates sizeof(`TYPE`) on stack
  - `... = malloc(BYTES)` → allocates `BYTES` on heap
  - `TYPE* NAME = malloc(sizeof(*NAME))` → allocates sizeof(void*) on stack (for the address variable) and sizeof(`TYPE`) on heap.
  - See reference sheet for more examples, e.g., data segment, BSS.
- Recursion
- Quiz #3 will be Wed 10/4

Note: Valgrind messages introduced on Fri 9/29, including “□ bytes after a block of size □”. See the malloc_buffer_overflow.c snippet from Fri 9/29 for more on that.
1 block → 1 call to malloc(⋯)

12 bytes was argument to malloc(sizeof(*q) + 3)

Hotel analogy
check in
check out

forget to checkout

too many people overflow the hot side room

going into someone else's room

malloc(⋯)
free(⋯)

int our platform:
4 bytes

leak
invalid write
invalid read
Inside a function

\[
\text{int} \ n = \ldots
\]

\[
\text{Integrand} \cdot a \_ \text{int}
\]

\[
\ldots = \text{malloc}(\ldots)
\]

\[
\text{int} \* a = \text{malloc}(\ldots)
\]

Outside a function, allocates on data segment or BS for globals
allocate space on STACK
allocates space on HEAP
on stack reserves space on heap