Last time:
- Memory
  - Stack vs. heap
- Pointers
- malloc
- How do we get data from one block of memory to another?

Today:
- Copying arrays (more generally, memory)
  - What happens to the old array?
- calloc
- How to think about pointers/malloc
- Resizing arrays
- Amortized analysis
- ADT teaser
```c
int main() {
    size_t arrlen = 4;
    int *arr = malloc(sizeof(int)*arrlen);
    arr[0] = 75;
    for (size_t i = 0; i < arrlen; i++) {
        arr[i] = i * 3;
    }
    printf("%p\n", (void*)arr);

    // Make arr bigger
    arrlen *= 2;
    arr = malloc(sizeof(int)*arrlen);
    printf("%p\n", (void*)arr);
}
```
```c
int f(int a, int b) {
    int x = a + b;
    return x - 1;
}
```

3

1. Allocate stack for 3 ints
2. Execute fn
3. Deallocate stack frame
1 + 2 + 4 + \ldots + 1000000ish

= 2\text{ million}

\text{where}

1 + 2 + 3 + \ldots + 1000000 = \frac{1000000^2}{2}
structured and has a stack implementation.

def f(x):
    head = Head(CircBuffer(x))
\( \text{arr} \rightarrow a = (*\text{arr}).a \)

\( \neq \quad *\text{arr}.a \)

\( 64 \)

\( \text{struct i} \)

\( \text{char c1j} \)

\( \text{char c2j} \)

1