



A terrible short story about me, Jerry “poohbear” Cain, and a mysterious, fearsome guardian of the heap called the Heap Alligator

Dynamic Allocation: Using the Heap

When we want memory on the heap, we use methods such as **malloc()** and **realloc()** to receive pointers to heap space allocated for us . We use **free()** to give it back.

```
char* annoy_jerry_str = malloc(8);
```

Though **malloc()** has a **void*** return type to denote an address to generic (i.e. any kind of) memory, we can set a pointer equal to without casting, e.g.,

```
> int* num_arr = malloc(n * sizeof(int));
> char** four_strs = malloc(4 * sizeof(char*));
> movie* beeMovie = malloc(sizeof(movie)); 
```

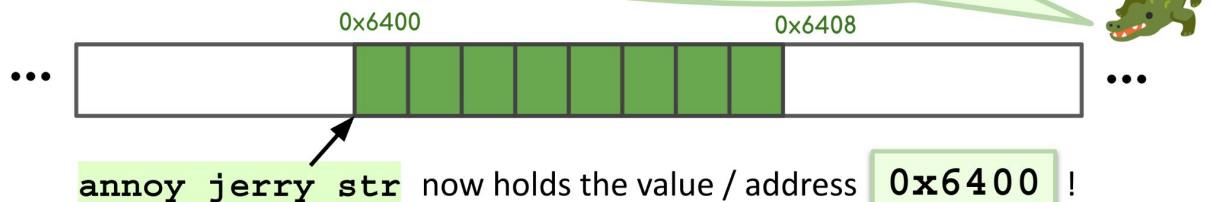
malloc to get pointer to heap memory

```
char* annoy_jerry_str = malloc(8);
```

Hey heap alligator, can I have 8 bytes of memory?

Sure thing! Here's 8 bytes starting at address **0x6400**.

Heap Alligator



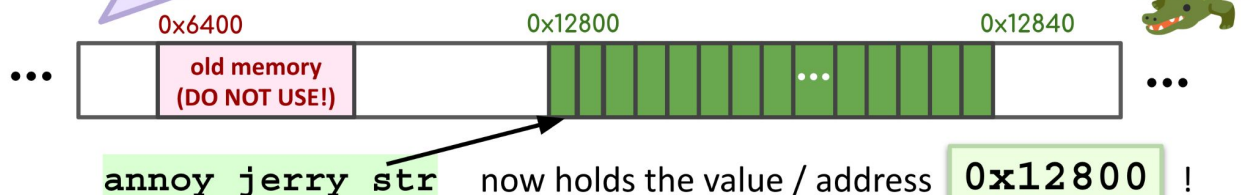
realloc to resize memory

Let's say later we want to upgrade our allocated space from 8 bytes up to 64 bytes!

```
annoy_jerry_str = realloc(annoy_jerry_ptr, 64);
```

Hey heap alligator, so sorry, but can I have 64 bytes of memory now? I need to annoy Jerry more :)

Ok ... sure, but I had to move you to address **0x12800** to have space.



Note: The pointer may stay in place, but sometimes it moves—depends on where space is available!



free to deallocate memory

Jerry is frowning 😞, so I should prob stop annoying him. Let's free the memory! We should make sure to **do this before the program ends** — otherwise a memory leak 😞

```
free(annoy_jerry_ptr);
```

Hey heap alligator, you can have the memory back! I'm done annoying Jerry (for today)

Sure Ben, thanks for being responsible! :)



The memory pointed to by `annoy_jerry_ptr` is now freed! 🦋

malloc, realloc may return NULL ptr ⚠️

When `malloc()`, `realloc()`, etc. can't fulfill our request, it returns a `NULL` pointer 🚫. We should check ⚠️ `assert(ptr != NULL);` after any request for heap memory.

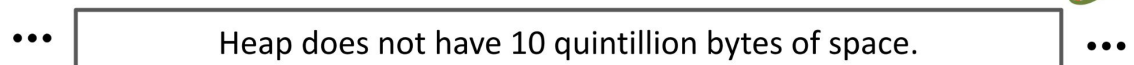
DO NOT DO THIS PLEASE LOL 🙅

```
annoy_jerry_ptr = malloc(LONG_LONG_MAX);
```

Hey heap alligator! I lied, so sorry. Can I have TEN QUINTILLION BYTES?? 🐱

What the BLEEP Ben.

Absolutely not.



Pointer `annoy_jerry_ptr` is assigned the error value **NULL**

✎ That day, Ben learned an important lesson about when poking the poohbear goes too far. For with the great power of the heap comes great ...

Stack vs Heap: Which to Use?

📈 What are the general **benefits of heap memory** over stack memory?

- Heap memory is **more plentiful** + we can **resize memory** after its initial allocation!
- **Heap memory persists outside a function call** (unlike stack memory, which is erased when the call is over), so we have better control over its memory lifetime.

📉 What are the general **disadvantages of heap memory** over stack memory?

- **More opportunities for errors** ⚠️, such as type errors, forgetting to free all allocations before the program is done, allocating wrong size, pointer errors, etc.
- **Stack allocation is faster, more convenient** (automatic allocation and deallocation, can declare and initialize in one step), and has superior type safety.

🎯 Programs typically use both in combination, but **generally stick to stack allocation unless we need dynamic allocation**, e.g., may need to resize later.