Rules for Dynamic Allocation

Be sure to follow the rules when using dynamically allocated memory:

1. Do not read/write memory locations before/after a block.
2. Call **`free`** exactly once on each block.
3. Do not call **`free`** on any other pointer, including pointers into a block.
4. Do not access (read, nor write) a block after freeing it.

Overloading Meaning: `realloc`

I mentioned earlier that one should avoid overloading function meaning for no reason.

`realloc` is a good example.

Can't remember **`malloc`**'s name?

Just use **`realloc (NULL, size)`**!

Can't remember **`free`**'s name?

Just use **`realloc (ptr, 0)`**!

File Scope Variables for Dynamic Resizing

Now we're ready to write code.

We will need some file-scope variables:

```
static player_t* player_list = NULL;
static int32_t num_players = 0;
static int32_t max_players = 10;
```

`player_list` is the array. We cannot statically initialize it to a dynamic block.

Write `player_create` Using Dynamic Resizing

We will write

```
int32_t player_create (char* n,
    char* pswd, int32_t p_age,
    player_t** new_p);
```

which **uses dynamic resizing**

- to **find a free array element**,
- **initialize it** using **`player_init`**, and
- **return a pointer in `*new_p`**.

The return value is **1 for success, 0 for failure**.

