~~Dynna~~

Dynamic Memory

opposite of this is

static memory

1. Does this work

2. Does ~~the~~ This work

3. Does this work ✓

Dynamic vs Static

$\uparrow$

pointers

```
int x[10];
```

```
int n;
scanf ("%d", &n);
int arr [n]; ← this is static
```

calloc, malloc, realloc, free

$\uparrow$ memory leaks otherwise

malloc   stdlib.h

(void *) malloc(unsigned long int);
size_t

create a block of memory on the heap

---

typedef
struct Point {

int x;

int y;

};

if you create a point you type:

struct Point p;

---

when you want to access memory in a point

Char *{name}; name 19 ⇒ size;

name = (char *) malloc((size + 1) * sizeof(char));

returns size_t

returns NULL when it fails.

x[5]   *(x+5);

x[0]   *(x+0) returns the first spot in the array

you should ~~[illegible]~~ always free memory allocated using malloc

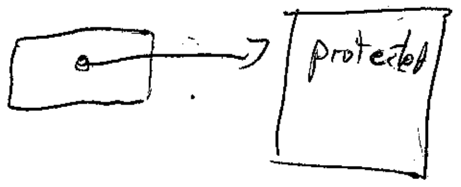
you can create a dangling pointer

Can result in 2 errors

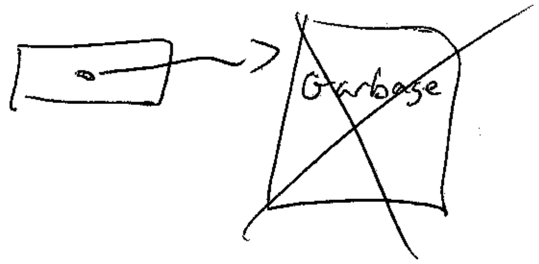
1. Use after free

2. Double free

before free



after free



~~Def~~

Other problems include dereferencing NULL

Don't change static memory either