Announcements

MP1, HW0 available, due 1/27, 11:59p.

Exam 1: 1/29-2/1, link to scheduler on website

Pointer variables and dynamic memory allocation:

```
int * p;
```

Stack memory

loc	name	type	value
a40	p	int *	

Heap memory

loc	name	type	value

Fun and games with pointers: (warm-up)

```
int * p, q;
```

What type is q?_______________

```
int *p;
int x;
p = &x;
*p = 6;
cout << x;
```

What is output?_______________

```
cout << p;
```

What is output?_______________

Write a statement whose output is the value of `x`, using variable `p`: ____________

```
int *p, *q;
p = new int;
q = p;
*q = 8;
cout << *p;
```
What is output?______________
```
q = new int;
*q = 9;
p = NULL;
```
Do you like this?_____________
```
delete q;
q = NULL;
```
Do you like this?_____________

Memory leak:

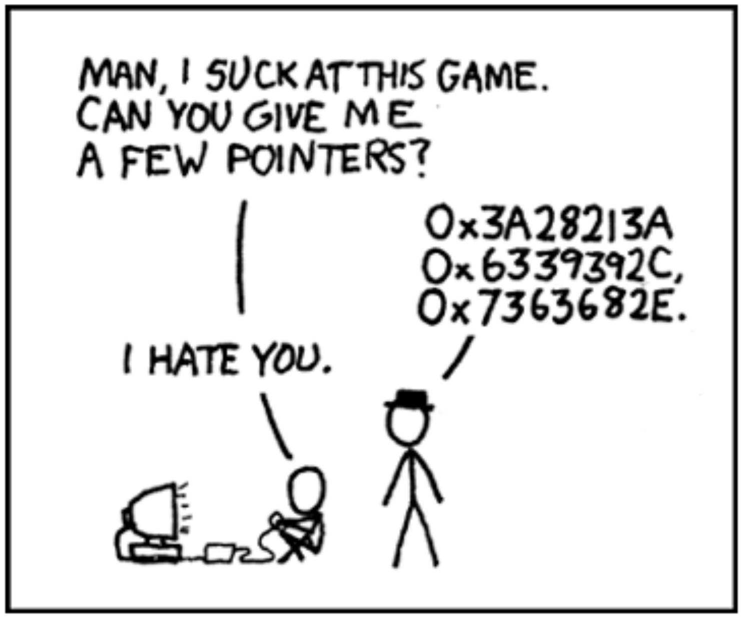
Deleting a null pointer:

Dereferencing a null pointer:

Fun and games with pointers:

```
int * p, * q;
p = new int;
q = p;
delete p;
… // some random stuff
cout << *q;
```

Do you like this?______________


MAN, I SUCK AT THIS GAME. CAN YOU GIVE ME A FEW POINTERS?
0x3A28213A 0x6339392C, 0x7363682E.
I HATE YOU.

Stack vs. Heap memory:

```
void fun() {
  string s = “hello!”;
  cout << s << endl;
}
int main() {
  fun();
  return 0;
}
```

System allocates space for s and takes care of freeing it when s goes out of scope.

Data can be accessed directly, rather than via a pointer.

```
void fun() {
  string * s = new string;
  *s = “hello?”;
  cout << *s << endl;
  delete s;
}
int main() {
  fun();
  return 0;
}
```

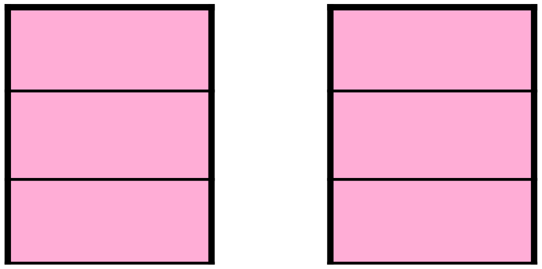
Allocated memory must be deleted programmatically.

Data must be accessed by a pointer.

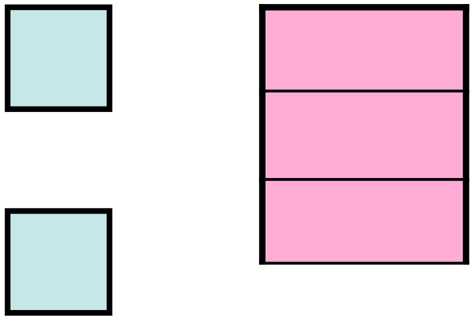
Pointers and objects:

```
class face {
public:
    void setName(string n);
    string getName();
    …
private:
    string name;
    PNG pic;
    boolean done;
};
```

```
face a, b;
… // init b
a = b;
a.setName("ann");
b.getName();
```



```
face * c, * d;
… // init *d
c = d;
c->setName("carlos");
(*d).getName();
```



Arrays: static (stackic)

```
int x[5];
```

Stack memory

loc	name	type	value

Arrays: dynamic (heap)

```
int * x;

int size = 3;
x = new int[size];

for(int i=0, i<size, i++)
    x[i] = i + 3;

delete [] x;
```

Stack memory

loc	name	value

Heap memory

loc	name	value