Announcements

MP2 available, due 9/15, 11:59p. EC: 9/8, 11:59p.

```
int *p, *q;
p = new int;
q = p;
*q = 8;
q = new int;
*q = 9;
p = NULL;
```

```
int *p, *q;
p = new int;
q = p;
*q = 8;
delete q;
*p = 12;
p = NULL;
```

```
int *p;
int x = 5;
p = &x;
delete x;
p = NULL;
```

```
int *p;
int x = 5;
*p = x;
```

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.

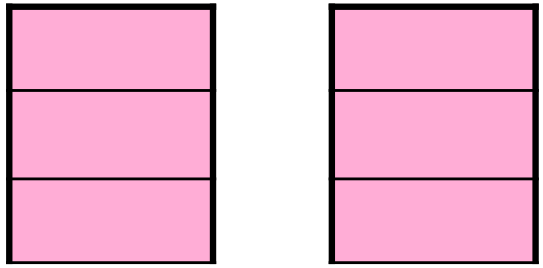
```
void fun() {
  string * s = new string;
  *s = "hello?";
  cout << *s << endl;
  delete s;
}
int main() {
  fun();
  return 0;
}
```

Allocated memory must be deleted programmatically.

Data must be accessed by a pointer.

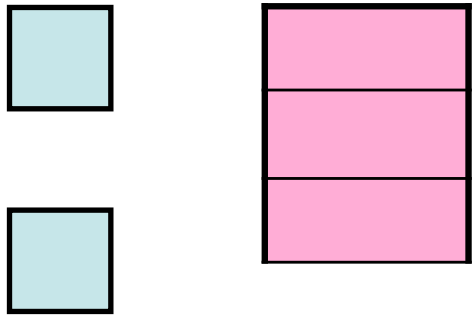
Pointers and objects:

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



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

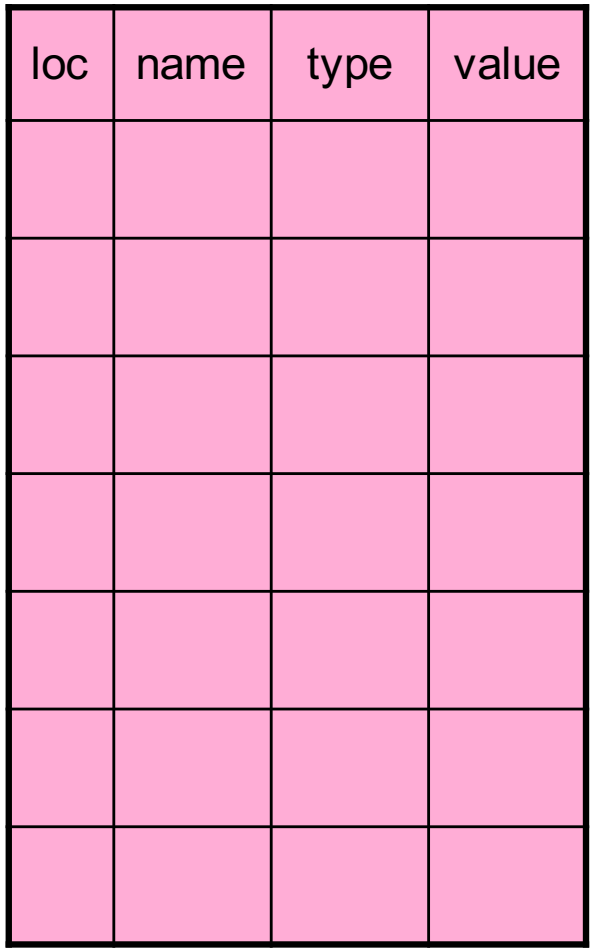
```
face * c, * d;
… // init *d
c = d;
c->setName("carlos");
(*d).getName();
```



Arrays: static (stackic)

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

Stack memory



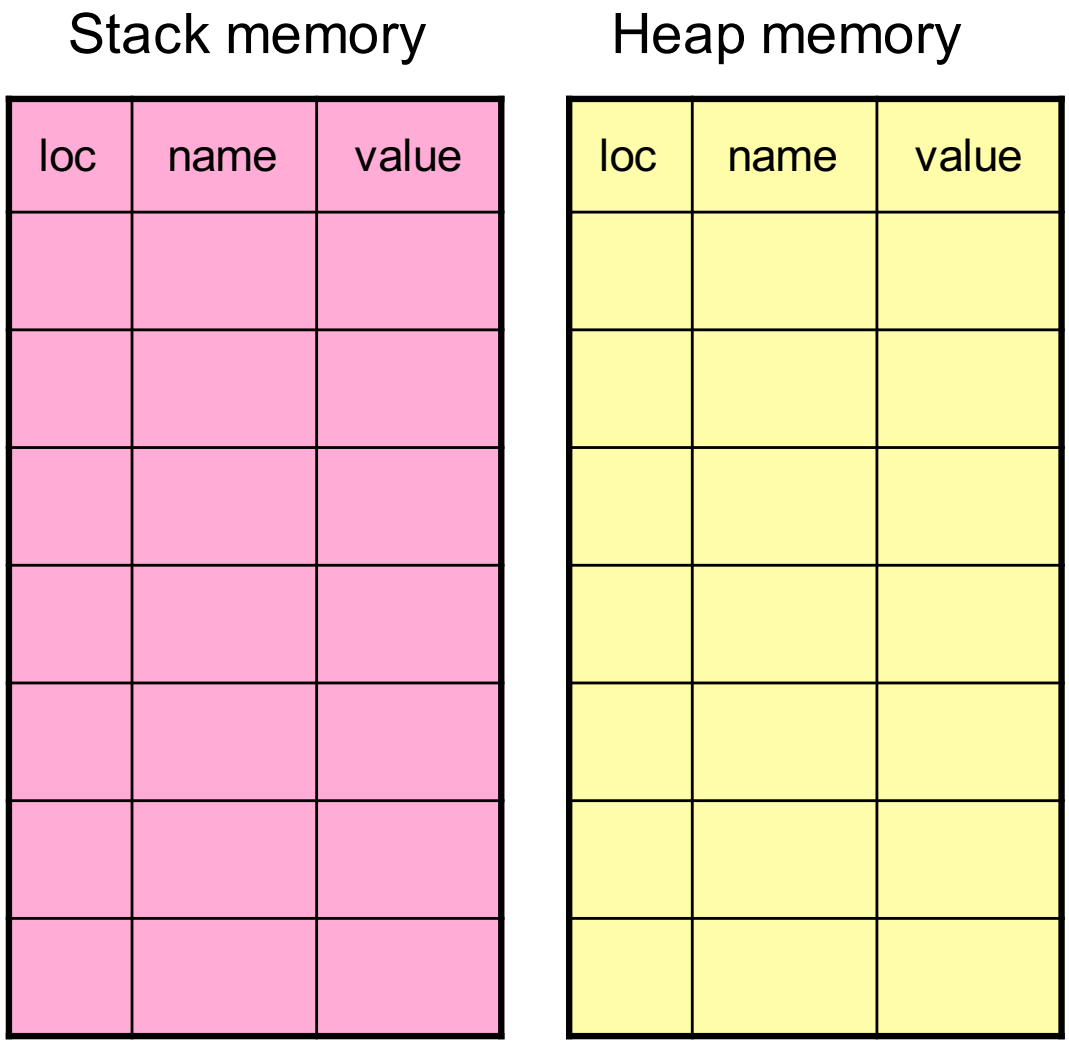
Arrays: dynamic (heap)

```
int * x;


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


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


delete [] x;
```



A point to ponder: How is my `garden` implemented?

```
class garden{
public:
…
// all the public members
…
private:
   flower ** plot;
   // other stuff
};
```

Option 1:

Option 2:

Option 3:

Option 4:

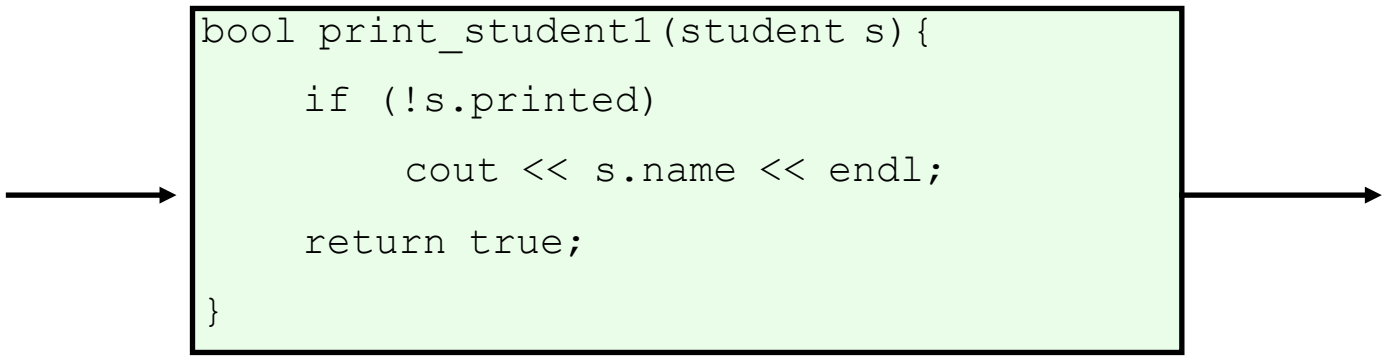
Parameter passing:

```
struct student {
    string name;
    PNG mug;
    bool printed; // print flag
};
```

What happens when we run code like this:

```
int main() {
    student a;
    print_student1(a);
}
```

?



Parameter passing:

```
struct student {
    string name;
    PNG mug;
    bool printed; // print flag
};
```

Function defn

```
bool print_student1(student s){
    if (!s.printed)
        cout << s.name << endl;
    return true;
}
```

Example of use

```
student a;
… // initialize a
a.printed = print_student1(a);
cout << a.printed << endl;
```