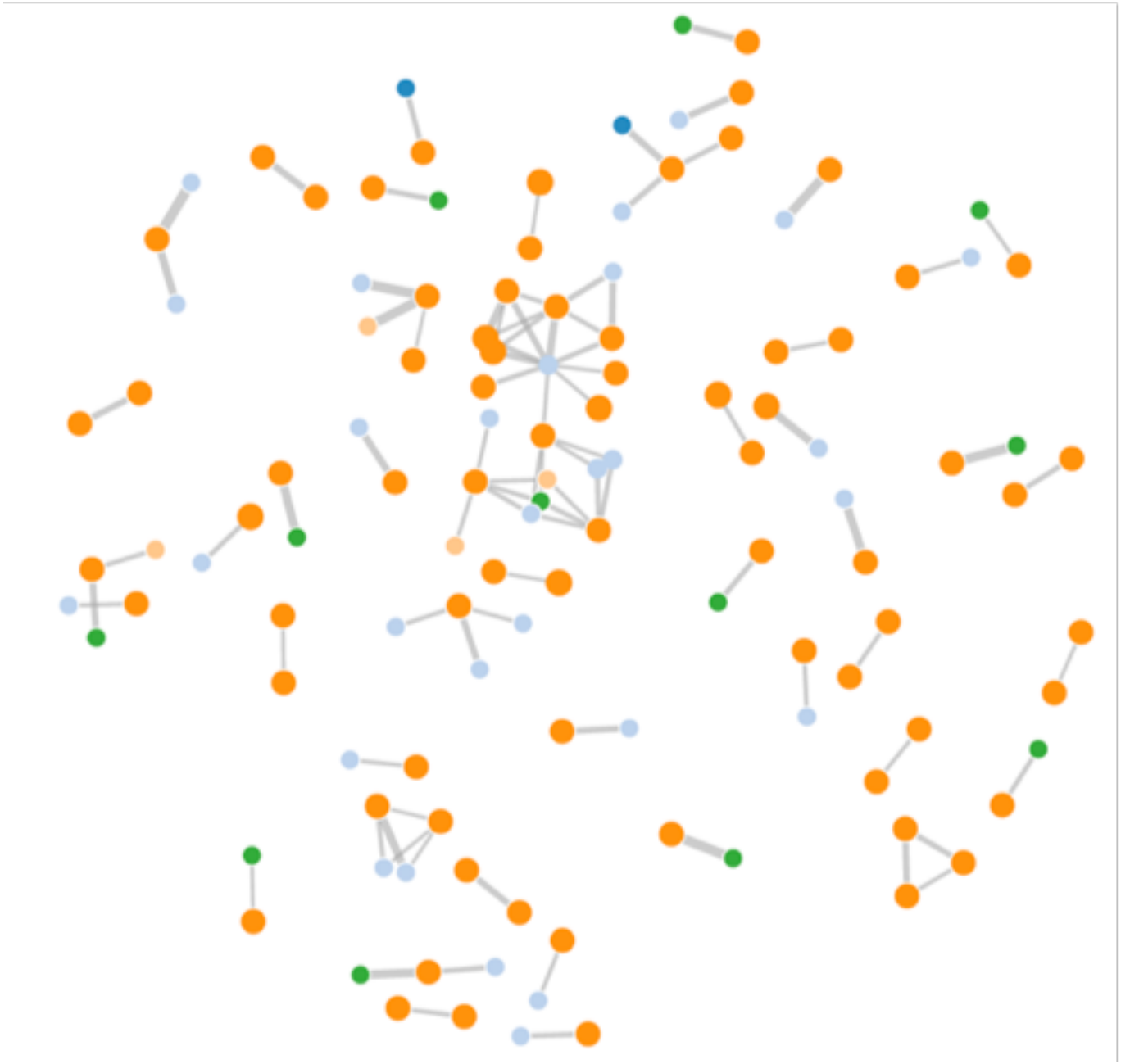


Announcements

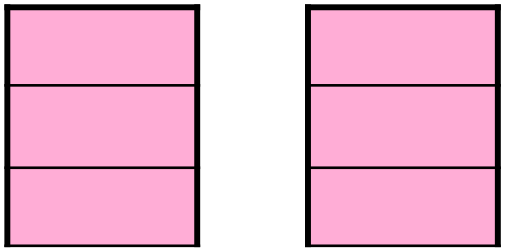
MP1, HW0 available, due 1/27, 11:59p.

Exam1: 1/29-2/1

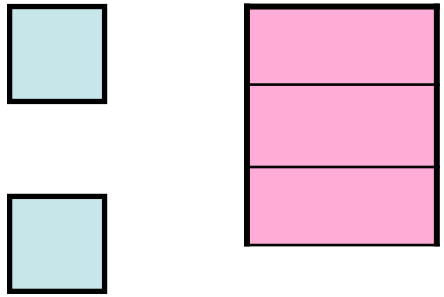
```
class face {
public:
     void setName(string n);
     string getName();
     …
private:
     string name;
     PNG pic;
     boolean done;
};
```

Pointers and objects:

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



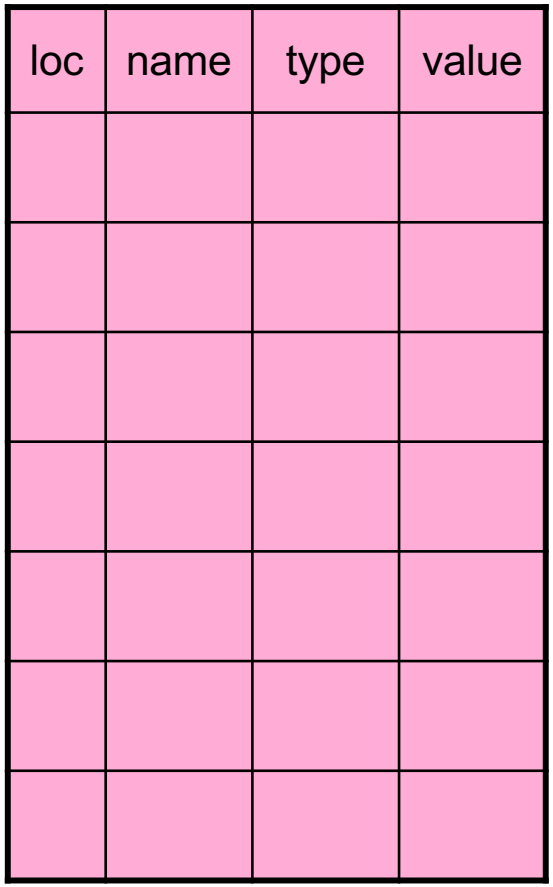
```
face * c, * d;
… // init *d
c = d;
(*c).setName("carlos");
d->getName();
```



Arrays: static (stackic)

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

Stack memory



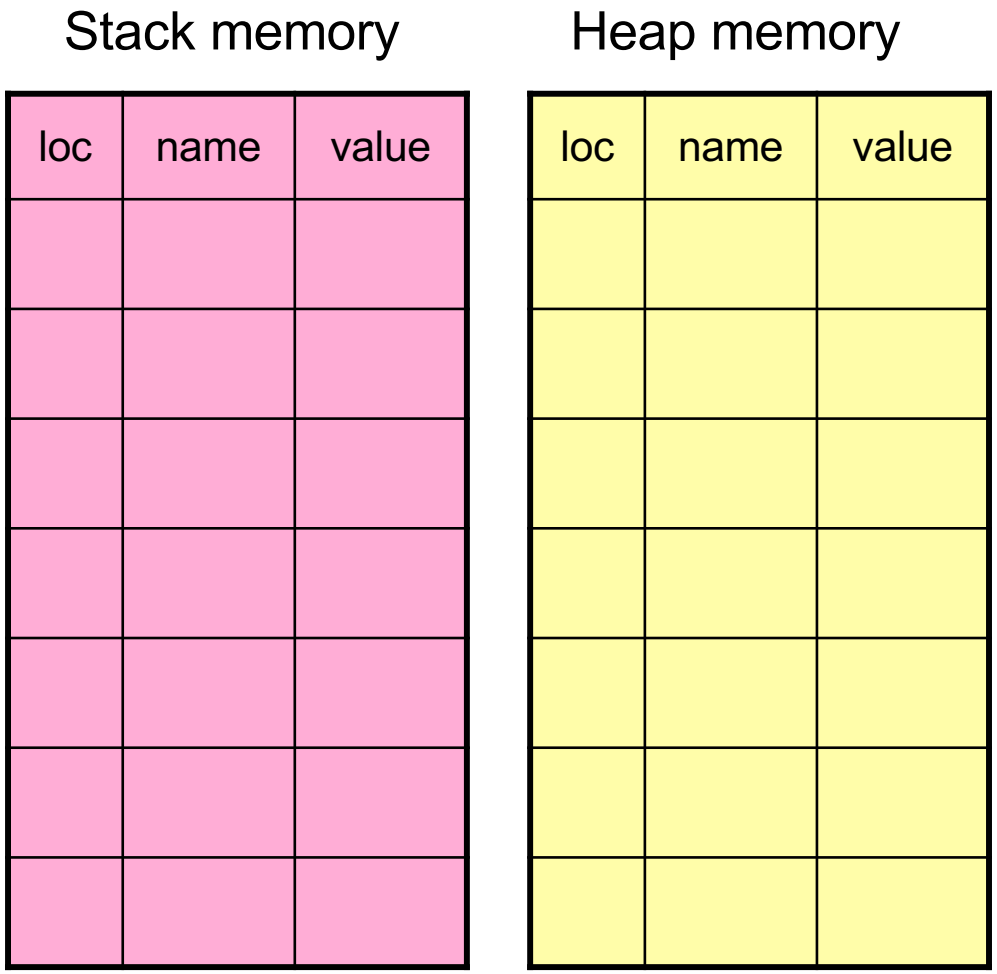
Arrays: dynamic (heap)

```
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:

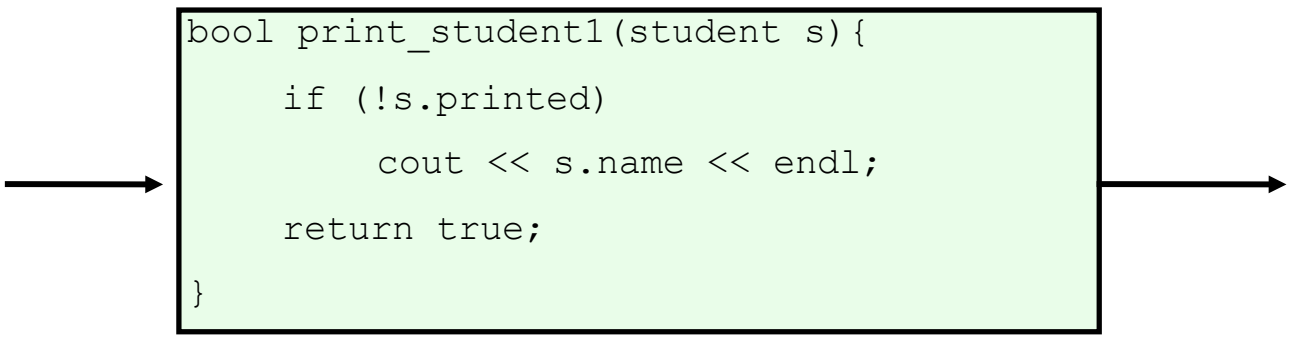
Parameter passing:

```
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:

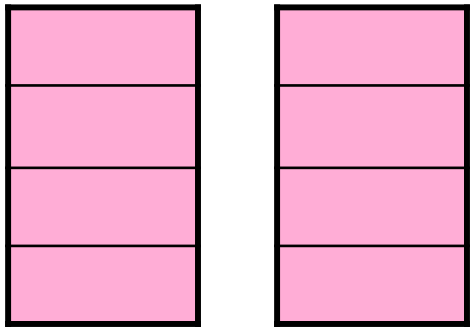
```
struct student {
    string name;
    PNG mug;
    bool printed; // print flag
};
```

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;
```



Parameter passing:

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

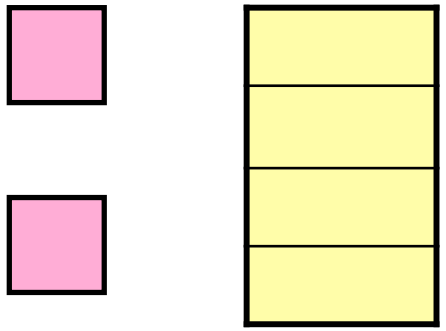
Function defn

```
void print_student2(student    s){
     if (!  s.printed)
          cout <<   s.name << endl;

}
```

Example of use

```
student * b;
… // initialize b
print_student2(b);
cout << b.printed << endl;
```



Parameter passing:

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

Function defn

```
void print_student3(student    s){
    if (!  s.printed)
        cout <<   s.name << endl;

}
```

Example of use

```
student c;
… // initialize c
print_student3(c);
cout << c.printed << endl;
```