CS 161
Intro to CS I
Arrays vs. Structs
How does creating and freeing memory work?

```cpp
int *r[5], **s;

for(int i=0; i < 5; i++)
  r[i]=new int;
for(int i=0; i < 5; i++)
  delete r[i];

for(int i=0; i < 5; i++)
  r[i]=new int[5];
for(int i=0; i < 5; i++)
  delete [] r[i];

s=new int*[5];
for(int i=0; i < 5; i++)
  s[i]=new int[5];
for(int i=0; i < 5; i++)
  delete [] s[i];
delete [] s;
```
Passing a 2-D Array (Dynamic)

```c
int main() {
    int **array; (or if you make int *array[2];
    ...
    pass_2darray(array);
    ...
}
void pass_2darray(int *a[]) {
    cout << "Array at zero: " << a[0][0] << endl;
}
OR
void pass_2darray(int **a) {
    cout << "Array at zero: " << a[0][0] << endl;
}
```
Create 2-D Array in Functions

```cpp
int main() {
    int **array;
    ...
    array = create_2darray(rows, cols);
    ...
}
int **create_2darray(int r, int c) {
    int **a;
    a = new int*[r];
    for(int i=0; i<r; i++)
        a[i] = new int[c];
    return a;
}
```
Create 2-D Array in Functions

```c
int main() {
    int **array;
    ...
    create_2darray(&array, rows, cols);
    ...
}
void create_2darray(int ***a, int r, int c) {
    *a = new int*[r];
    for(int i=0; i<r; i++)
        (*a)[i] = new int[c];
}
```
Create 2-D Array in Functions

```c
int main() {
    int **array;

    ... create_2darray(array, rows, cols);
    ...
}
void create_2darray(int **&a, int r, int c) {
    a = new int[r];
    for(int i=0; i<r; i++)
        a[i] = new int[c];
}
```
Structures

• Data Structures So Far...
  – Variables
  – Arrays

• What if we want mixed types?
  – Record: name, age, weight, etc.
  – Use **struct** type
Why is it good to have an array of structs?

• What happens if you have two arrays with first names and last names, and you want to sort by first name?

• What happens if you put the first name and last name in a struct?
struct record {
    char name[50];
    int age;
    float weight;
};

• What does this do?
• How do we use it?
Struct Type

```c
struct doc_record{
    char name[50];
    int age;
    float weight;
}; //creates a user defined type, doc_record
int main() {
    doc_record garrett; //use it as a type
    ...
}
```
Creating Struct Demo...
struct contact_info {
    std::string name;
    std::string address;
    unsigned int phone;
};
...
int main() {
    contact_info address_book[50];
    ...
    address_book[0] = create_contact();
    ...
}
contact_info create_contact() {
    contact_info contact;
    contact.name = “Jennifer”;
    return contact;
}
What about passing structs to functions?

```cpp
#include <iostream>
#include <cstring>
using namespace std;

struct contact {
    string name;
    string address;
    string phone;
};

//Pass struct by reference to change value
void set_name(contact &c) {
    c.name = "jennifer";
}

int main() {
    contact address_book[2]; //Create an array of contacts
    //address_book[0].name = "jennifer";
    //Pass 1st contact in book
    set_name(address_book[0]);
    cout << address_book[0].name << endl;
    return 0;
}
```
What about passing structs to functions?

```cpp
#include <iostream>
#include <cstring>
using namespace std;

struct contact {
    string name;
    string address;
    string phone;
};

//Pass struct by pointer to change value
void set_name(contact *c) {
    c->name = "jennifer";
}

int main() {
    contact address_book[2]; //Create an array of contacts

    //address_book[0].name = "jennifer";

    //Pass address of first contact in book
    set_name(&address_book[0]);

    cout << address_book[0].name << endl;
    return 0;
}
```

Structs vs. Classes

• Structs only have state/attributes/member variables
• Classes have state/attributes/member variables

PLUS
• Classes have behavior/member functions