# APS105 Lecture 26
# Quicksort, 10.1

Dan Zingaro

March 26, 2010

# ConcepTest

```
int partition (int low, int high, int pivot, int a[]) {
  int i = low, j = low;
  while (j <= high) {
    if (a[j] < pivot) {
      swap (a, i, j);
      i++;
    }
    j++;
  }
  return i;
}
```

Assume we partition the array `[6, 8, 3]` around pivot 4. What is the resulting array?

- A. `[3, 6, 8]`
- B. `[8, 6, 3]`
- C. `[3, 8, 6]`

# Feedback from Reading Quiz

```
struct date {
  int year, month, day;
};
struct date *d;
(*d).year = 2010; /*problem!*/
```

- "Do you recommend that we use `typedef` to create synonyms for our structure?"
  - You can, but we won't study `typedef` in this course
- "How is the struct helping us in lab5?"
  - Using a `struct` is another way to return multiple values from a function

# What are Structures?

| | Data must be same type? | Creation | Access |
|---|---|---|---|
| Array | Yes | `[] syntax` | `a[index]` |
| Structure | No | `struct...` | `s.name` |

The following `struct` has three members.

```
struct student {
  char firstName[20];
  char lastName[20];
  int year;
};
```

# Declaring Structure Variables

- Now, we can use `struct student` as a type, just like `int` or `float`
- Careful: it's `struct student`; using `student` by itself is incorrect
- To make some students and modify their members ...

```
struct student student1, student2;
student1.firstName = "Dan";
student1.lastName = "Jones";
student1.year = 3;
...
```

# ConcepTest

```
struct name {
  char firstName[20];
  char lastName[20];
};

struct student {
  struct name studentName;
  int year;
};

struct student s;
```

How do I change the first letter of the first name of s?

- A. `s.name.firstName[0] = 'D';`
- B. `s.studentName.firstName[0] = 'D';`
- C. `s.studentName[0] = 'D';`
- D. `s.firstName[0] = 'D';`

# Structs and Functions

- Functions can take structs as parameters, and return a struct back to the caller

```
void printStudent (struct student s) {
  printf ("First name: %s\n", s.firstName);
  printf ("Last name: %s\n", s.lastName);
  printf ("Year: %d\n", s.year);
}
```

# ConcepTest

```
struct student {
  char firstName[20];
  char lastName[20];
  int year;
};
```

Consider the following function prototype:

```
void proc (struct student s1, struct student s2, char *p);
```

Also assume that `a` and `b` have been declared as `struct student` and have been initialized.
Without worrying about what the function will do, which function call is valid?

- A. `proc (a, b, "go");`
- B. `proc (a, a, "go");`
- C. `proc (a, a, b.firstName);`
- D. A and B
- E. All of the above