Structs

What's A Struct?

A struct is a place to store related data. Think about a student, they have various pieces of data about them that are related:

- id – int
- age – int
- name – string
- gpa – double

There are all these attributes, but using a struct we can put them all in the same place.

Defining and Declaring a Struct?

There are a few ways to declare a struct:

Method 1:

Define:

```
     struct sampleStruct
     {
          int a;
          double b;
     };
```

Declare:

```
     struct sampleStruct myStruct; //myStruct is the name for this
variable
```

Method 2:

Define:

```
     struct sampleStruct
     {
          int a;
          double b;
     };
     typedef struct sampleStruct a_struct;
     //Here "a_struct" is just another name,
     //you can use whatever you want.
```

Declare:

```
     a_struct myStruct; //Notice you don't need the "struct" keyword.
```

Method 3 (my favorite):

Define:

```
     typedef struct
     {
          int a;
          double b;
```

```
    } sampleStruct;
```

Declare:

```
    sampleStruct myStruct;
```

Accessing Members

You can use the dot ('.') operator to access the “members” of the struct.

```
        sampleStruct myStruct;
        myStruct.a = 1;
        myStruct.b = 5.432;
        printf(“%f\n”,  myStruct.a +  myStruct.b);
        scanf(“%d”,  &(myStruct.a));
        //The parens arount  “myStruct.a” are not necessary here,
        //but I like them
```

Passing as Parameters

After you have defined a struct it is just like any other datatype. Therefore, you can pass it just like any other type.

```
        void printStruct(sampleStruct passedStruct)
        {
            printf(“%d, %f\n”, passedStruct.a, passedStruct.b);
        }
```

Arrays of Structs

Since structs can be used like normal datatypes, you can make arrays of them!

```
        sampleStruct arr[10];
        arr[0].a = 5; //Note how I first index into the array and
        then use the dot.
        arr[0].b = 2.1;
```

Casting

If you are felling particularly adventurous you can cast structs just like you cast primitives (int, double, etc...). But be careful, they structs better be the same.

```
                typedef struct      typedef struct
                {                   {
                    int a;              int a;
                    double b;           double b;
                } sampleStruct;     }
                                    anotherStruct;

        sampleStruct struct1;
        anotherStruct struct2;

        struct1.a = 5;
        struct1.b = 1.2;
```

```
struct2 = (anotherStruct)struct1; //WORKS!
```

Questions

1. Write a struct that has a string as a member.

Program

Write a program that holds data for a certain amount of students and then prints it all out.

Specs

1. Students have the following properties:
    1. first name (one word)
        1. This will be at most 20 chars.
    2. gpa
    3. age
2. Take in all the names at on time.
3. Take in all the gpas at one time.
4. Take in all the ages at one time.
5. Assume there will be 3 students, but use a CONSTANT.
6. After you get all the information, print out each student.

Test Runs (input in **bold**)

```
First name for all students: a b c
Age for all students: 1 2 3
GPA for all students: 1.1 2.2 3.3
Student #0: name: a, age: 1, gpa: 1.100000
Student #1: name: b, age: 2, gpa: 2.200000
Student #2: name: c, age: 3, gpa: 3.300000

First name for all students: i see howItIs
Age for all students: 12 43 -12
GPA for all students: -234 23424.234 0
Student #0: name: i, age: 12, gpa: -234.000000
Student #1: name: see, age: 43, gpa: 23424.234000
Student #2: name: howItIs, age: -12, gpa: 0.000000
```