What is a Structure?

- Compound data.

- Name: Coolio McCool
  Date of Birth: 01.01.01
  Student ID: 0101010010

Structures in Racket vs. C

Structures in C are similar to structures in Racket:

Racket:

```racket
(struct posn (x y)
 #:transparent)
(define p (posn 1 2))
(define a (posn-x p))
(define b (posn-y p))
```

C:

```c
struct posn {
  int x;
  int y;
};
const struct posn p = {1,2};
const int a = p.x;
const int b = p.y;
```

Racket generates selector functions when you define a structure. Instead of selector functions, C has a `structure operator (.)` which selects the value of the requested field.
# Structure Coding Example in C

```c
#include "cs136.h"

struct mystruct{
    int number;
    char letter;
};

int main(){
    struct mystruct s = {5,'a'};
    printf ("my struct %d, %c \n", s.number, s.letter);
    s.number = 6;
    s.letter = 'b';
    printf ("my struct %d, %c \n", s.number, s.letter);
    return 0;
}
```

**Practice Problem: Structure – collinear**

On Seashell, we have given you the structure `posn`:

```c
struct posn {
    int x;
    int y;
};
```

Implement the function `collinear(a, b, c)` which consumes three `posns` and returns true if a, b and c are collinear, otherwise returns false.

---

**Call Stack and Stack Frames**

Suppose the function `main` calls `f`, then `f` calls `g`, and `g` calls `h`.

As the program jumps from function to function, we need to remember the history of the return addresses, as well as all the parameters and local variables. This history is known as the **call stack**.

The entries that are pushed onto the call stack are known as **stack frames**.
Each function call creates a new stack frame that contains the following:

// ===========================
// <function name>:
// <parameter one>: <value>
// <parameter two>: <value>
// ...
// <local variable one>: ( <value> || ??? )
// <local variable two>: ( <value> || ??? )
// ...
// return address: <caller function name>:<line>
// ===========================

If a question asks you to draw the call stack, then for each stack frame pushed onto the call stack you must provide the above.

For example:

```c
int g(void){
    int x = 2;
    return x;
}

int f(int x){
    int a = g();
    return x * a;
}

int main(void){
    int x = 4;
    int a = 1;
    int y = f(x) + a;
}
```

Example: draw the call stack

```c
1 int g(void){
2    int x = 2;
3    return x;
4 }
5
6 int f(int x){
7    int a = g();
8    return x * a;
9 }
10
11 int main(void){
12    int x = 4;
13    int a = 1;
14    int y = f(x) + a;
15 }
```

```
main:
  x: 4
  a: 1
  y: ?
return address: OS
```

CS 136 Spring 2018 Tutorial 5 9
example: draw the call stack

```c
int g(void){
    int x = 2;
    return x;
}

int f(int x){
    int a = g();
    return x * a;
}

int main(void){
    int x = 4;
    int a = 1;
    int y = f(x) + a;
}
```

---

Example: draw the call stack

```c
int g(void){
    int x = 2;
    return x;
}

int f(int x){
    int a = g();
    return x * a;
}

int main(void){
    int x = 4;
    int a = 1;
    int y = f(x) + a;
}
```

---

Example: draw the call stack

```c
int g(void){
    int x = 2;
    return x;
}

int f(int x){
    int a = g();
    return x * a;
}

int main(void){
    int x = 4;
    int a = 1;
    int y = f(x) + a;
}
```
example: draw the call stack

```c
int g(void){
    int x = 2;
    return x;
}

int f(int x){
    int a = g();
    return x * a;
}

int main(void){
    int x = 4;
    int a = 1;
    ⇒ int y = f(x) + a;
    return 0;
}
```

---

example: structures on the call stack

```c
struct foo {
    int x;
    int y;
};

int baz(struct foo qux) {
    ⇒ return qux.x + 1;
}

int main(void) {
    struct foo bar = {5, 6};
    int x = baz(bar);
    return 0;
}
```
Online Tracing

- CS 136 has an online program tracer.
- Contains many useful examples.
- Includes stack frames (and even heap!).
- https://www.student.cs.uwaterloo.ca/cs136/examples/