CSCB09H Worksheet: structs

1. Here is the beginning of a program involving structs. You will need to fill in missing bits. If you can work with a partner with a machine and actually compile your program at each step, do that. If not, it will be fine to work on paper.

```
#define MAX_NAME_SIZE 32
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

struct player {
   char name[MAX_NAME_SIZE];
   char *position;
   int home_runs;
   float avg;
};

int main() {
    // Declare a struct player called p1.




    // Initialize it to represent the third baseman Josh Donaldson,
    // whose batting average is 0.297. He has hit 41 home runs.
```

2. Here we have added a declaration for a pointer to a struct player. Allocate space for the struct on the heap. Remember to initialize `p2` so that it can refer to this memory.

   Error-checking: Check if the memory allocation was successful (how?), and if it failed, exit the program with a non-zero exit status.

```
    struct player *p2;
```

CSCB09H Worksheet: structs

3. Write a function `out_of_the_park` that increments the home-run count for the player passed as the function's argument. Think carefully about what the type of the function parameter should be.

4. Show how to make calls to `out_of_the_park` using `p1` and `p2`.

5. Suppose we have the following function declaration.

   ```
   void f(struct player p) { // Body hidden }
   ```

   Now suppose we call it from `main` using `f(p1)`. Draw the memory diagram of the program immediately after `f` is called, but before it returns. For extra practice, include `p2` and related memory in your diagram.

6. Something to think carefully about: can the body of `f` affect the local `p1` of `main`? In other words, after `f(p1)` exits, can any data associated with `p` have changed?

7. On a new sheet of paper, repeat the previous two questions when you call `f(*p2)` instead.