Quiz 3

CSCI 331: Fall 2021

Your name: _____

Memory Allocation Quiz

Refer to the buggy code below to answer the questions.

```
1  #include <stdio.h>
2
3  typedef struct record {
4    char first_name[20];
5    char last_name[20];
6    char identifier[20];
7  } record_t;
8
9  record_t* create_record(char* fname, char* lname, char* id) {
10   record_t r;
11   r.first_name = fname;
12   r.last_name = lname;
13   r.identifier = id;
14   return &r;
15 }
16
17 int main() {
18   record_t* r = create_record("Jason", "Bourne", "70014051");
19   printf("First name: %s\n", r->first_name);
20   printf("Last name: %s\n", r->last_name);
21   printf("Identifier: %s\n", r->identifier);
22
23   return 0;
24 }
```

Answer the following questions.

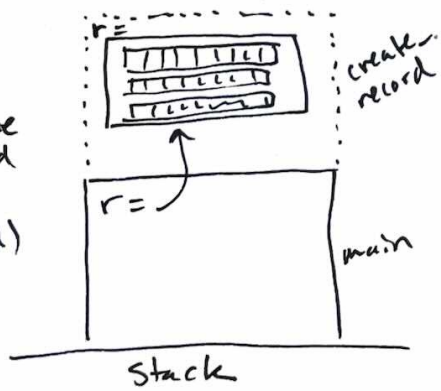
1. This program has several bugs. What are they?

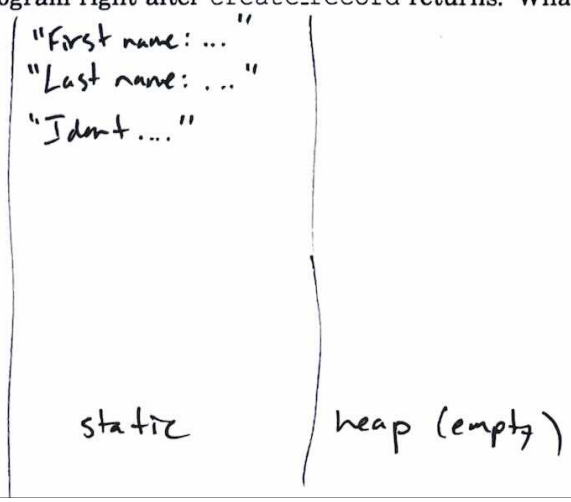
Bug #1: create_record returns a pointer to a local (stack-allocated) record; it is no longer valid in main (i.e., a "use-after-free" bug).

Bug #2: Lines 11–13 attempt to assign pointers to ~~statically~~ locally-allocated char arrays. This is a type error. Use strcpy instead.

2. Draw the memory state of the program right after `create_record` returns. What problems do you see?

r in main points to stack frame of returned function (create_record)





3. How might I fix this program? Write your modified code below.

one solution...

```
#include <stdio.h>
typedef struct record {
    char first_name [20];
    char last_name [20];
    char identifier [20];
} record_t;

record_t* create_record(char* fname, char* lname, char* id){
    record_t* r = malloc(sizeof(record_t));
    if(!r){
        fprintf(stderr, "Could not allocate record.\n");
        exit(1);
    }
    strcpy(r->first_name, fname);
    strcpy(r->last_name, lname);
    strcpy(r->identifier, id);
    return r;
}

int main() {
    record_t* r = create_record("Jason", "Bourne", "1234");
    printf("First name: %s\n", r->first_name);

    free(r);
    return 0;
}
```

(also, technically, stdlib.h for malloc and string.h for strcpy…)

Alternatively, stack allocate a record_t in main, and alter create_record to take a pointer to a buffer (which holds the stack allocated record_t). No need to free in that case.