

Lab 3.5 Worksheet

1. Boolean Problems

- a. Does the following C program compile? If it does, you don't need to explain why. If it doesn't, explain why and propose a change that would fix the compilation issue.

```
#include <stdio.h>

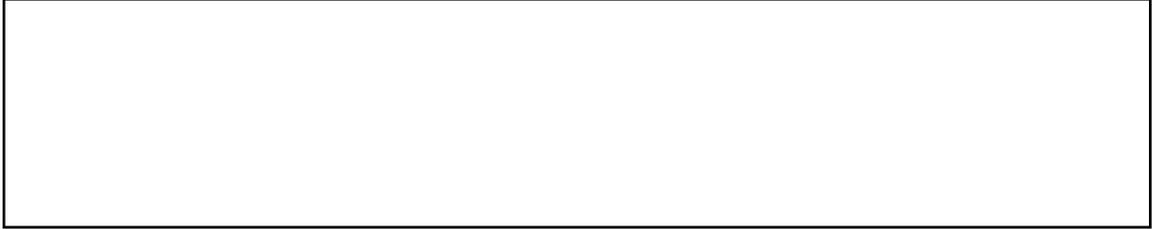
int main() {
    welcome("Chippy", 4);
}

void welcome(char* name, int num) {
    printf("Welcome to Lab %d, %s!\n", num, name);
}
```

- b. Does the below assertion *always* hold? Explain why or why not.

```
#include <stdint.h>
#include <assert.h>

int main() {
    int8_t num = 0;
    for (int i = 0; i < 128; i++) {
        num++;
    }
    assert (num == -128);
}
```



2. Memory Troubles

- a. Chippy, the class mascot, has recently started interning for Rainforest, a successful online retailing company. To improve shipping, Chippy has written the C function `packagify` that takes information about a package and bundles it up into a `struct`.

```
#include <stdlib.h>
#include <string.h>

typedef struct {
    int id;
    float longitude;
    float latitude;
    char* status;
} package_t;

package_t* packagify(int id, float longitude, float latitude, int status) {
    package_t package;
    package.id = id;
    package.longitude = longitude;
    package.latitude = latitude;
    switch (status) {
        case 0:
            package.status = malloc(12 * sizeof(char));
            memcpy(package.status, "Not shipped");
            break;
        case 1: package.status = "In transit"; break;
        case 2: package.status = "Delivered"; break;
    }

    return &package;
}
```

However, the lead engineer on Chippy's team has found several problems with `packagify`. What are these issues? At a high level, how can these issues be resolved? There may be multiple solutions.

- b. Your friend is trying their hand in the art of pointer magic and has written up the following C program. When you ask them to explain the code, they just enigmatically say that “a magician never reveals their secrets.”

What do the functions `abracadabra` and `hocuspocus` do? Before the program exits, what is the state of `matrix`?

```
void abracadabra(int** matrix, int i, int n) {
    int* p1 = *(matrix + i);
    int* p2 = p1 + (n - 1);
    int temp = *p1;

    *p1 = *p2;
    *p2 = temp;
}

void hocuspocus(int** matrix, int i, int n) {
    int** p1 = matrix + i;
    int** p2 = matrix + (n - 1);
    int* temp = *p1;

    *p1 = *p2;
    *p2 = temp;
}

int main() {
    int r1[] = {1, 2, 3, 4};
    int r2[] = {5, 6, 7, 8};
    int r3[] = {9, 10, 11, 12};
    int r4[] = {13, 14, 15, 16};
    int* matrix[4] = {r1, r2, r3, r4};

    hocuspocus(matrix, 0, 4);
    abracadabra(matrix, 3, 4);
    abracadabra(matrix, 2, 4);
    hocuspocus(matrix, 1, 4);
}
```

