CS 241 - Intro to C

Welcome to CS 241! You will learn a lot of new and exciting topics. This course will challenge you in ways you never knew, and ultimately make you a better programmer.

Warm Ups - macros/sizeof

What does the following code print?

```c
int a = 0;
size_t a_size = sizeof(a++);
printf("size: %zu, a: %d\n", a_size, a);
```

**Why does the code print that? Is the sizeof operator actually being evaluated?**

Is something wrong with the following snippet?

```c
#define swap(a, b) temp = a; 
    a = b; 
    b = temp;
```

```c
void selection_sort(int* a, size_t len) {
    size_t temp = len - 1;
    for (size_t i = 0; i < temp; ++i) {
        size_t min_index = i;
        for (size_t j = i+1; j < len; ++i) {
            if(a[j] < a[i]) min_index = j;
        }
        if (i != min_index)
            swap(a[i], a[min_index]);
    }
}
```

Give an English description for why the code doesn’t work. What should you look out for with preprocessors? (That’s why we try to use them sparingly.)

A Bit about Bits

C is one of the lowest-level languages most programmers will ever have to use. Draw out what the following code does:

```c
short mystery_bits(short input) {
    short max_set = ~0;
    short masked = input & (0xFF00 ^ max_set);
    short shifted = masked << 8;
    short ret = (shifted | 0xCC);
    return ret;
}
```

Fill out the bytes in the table (input is already filled out for you):

<table>
<thead>
<tr>
<th>Variable</th>
<th>Byte 1 (most significant)</th>
<th>Byte 2 (least significant)</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>CA</td>
<td>FE</td>
</tr>
<tr>
<td>max_set</td>
<td></td>
<td></td>
</tr>
<tr>
<td>masked</td>
<td></td>
<td></td>
</tr>
<tr>
<td>shifted</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ret</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


When does the following code not work?

```c
void positive_under_ten(int input) {
    if (0 < input < 10) {
        printf("Input is in the range\n");
    } else {
        printf("Input is not in the range\n");
    }
}
```

List out the order of operations when input = -1, 3, and 20:

Here’s an (incorrect) example for input = -1:

1. The if is evaluated. The condition 0 < input < 10 is evaluated and returns false.
2. The if statement is false, so it’ll jump to the else, and “Input is not in the range is printed”.

What is wrong with the following switch-case code?

```c
int print_error(int err_num) {
    switch(err_num) {
    case ENOENT:
        printf("No such file or entry\n");
    case EINTR:
        printf("Interrupted\n");
    default:
        break;
    }
}
```

What does break actually mean? When is it used?