1 Pre-Check

This section is designed as a conceptual check for you to determine if you conceptually understand and have any misconceptions about this topic. Please answer true/false to the following questions, and if false, correct the statement to make it true:

1.1 True or False: C is a pass-by-value language.

1.2 The following is correct C syntax:
   ```c
   int num = 43
   ```

1.3 In compiled languages, the compile time is generally pretty fast, however the runtime is significantly slower than interpreted languages.

1.4 The correct way of declaring a character array is `char[]` array.

1.5 Bitwise and logical operations result in the same behaviour for given bitstrings.

2 Bit-wise Operations

In C, we have a few bit-wise operators at our disposal:

- **AND (\&)**
- **NOT (\~)**
- **OR (|)**
- **XOR (\^)**
- **SHIFT LEFT (\ll)**
  - Example: `0b0001 \ll 2 = 0b0100`
- **SHIFT RIGHT (\gg)**
  - Example: `0b0100 \gg 2 = 0b0001`
C and Pointers

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>a &amp; b</th>
<th>a</th>
<th>a &amp; b</th>
<th>~a</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

For your convenience, truth tables for the logical operators are provided above. With the binary numbers a, b, and c below, perform the following bit-wise operations:

a = 0b1000 1011
b = 0b0011 0101
c = 0b1111 0000

(a) a & b
(b) a \land c
(c) a | 0
(d) a | (b >> 5)
(e) \sim ((b | c) & a)

3 Pass-by-who?

3.1 Implement the following functions so that they work as described.

(a) Swap the value of two int. **Remain swapped after returning from this function.**
   Hint: Our answer is around three lines long.
   ```c
   void swap(int, int) {
   }
   ```

(b) Return the number of bytes in a string. **Do not use strlen.**
   Hint: Our answer is around 4 lines long.
   ```c
   int mystrlen(char str) {
   }
   ```
4 Debugging

4.1 The following functions may contain logic or syntax errors. Find and correct them.

(a) Returns the sum of all the elements in `summands`.

```c
int sum(int *summands) {
    int sum = 0;
    for (int i = 0; i < sizeof(summands); i++)
        sum += *(summands + i);
    return sum;
}
```

(b) Increments all of the letters in the `string` which is stored at the front of an array of arbitrary length, \( n \geq \) `strlen(string)`. Does not modify any other parts of the array's memory.

```c
void increment(char *string, int n) {
    for (int i = 0; i < n; i++)
        *(string + i)++;
}
```

(c) Copies the string `src` to `dst`.

```c
void copy(char *src, char *dst) {
    while (*dst++ = *src++);
}
```

(d) Overwrites an input string `src` with "61C is awesome!" if there's room. Does nothing if there is not. Assume that `length` correctly represents the length of `src`.

```c
void cs61c(char *src, size_t length) {
    char *srcptr, replaceptr;
    char replacement[16] = "61C is awesome!";
    srcptr = src;
    replaceptr = replacement;
    if (length >= 16) {
        for (int i = 0; i < 16; i++)
            *srcptr++ = *replaceptr++;
    }
}
```