Part I: Bitwise operators and Bit-masking

C provides bitwise commands for AND(&), OR(|), XOR(^), and NOT(~). Ignoring NOT for now, let's see what happens when we reduce the 2-input gates to 1-input gates by fixing the second input.

a. Let x be the input. Fill in the following blanks with either 0, 1, x, or \(\overline{x}\) (NOT x):

\[
\begin{align*}
    x \& 0 &= _0 \\
    x \& 1 &= _x \\
    x | 0 &= x \\
    x | 1 &= 1 \\
    x ^ 0 &= _x \\
    x ^ 1 &= \overline{x}
\end{align*}
\]

b. Based on your responses, look at the columns (grouped by operation) above. Which operation would be useful for turning bits OFF? For turning bits ON? For flipping bits?

\& 0 turns off, | 1 turns on, ^ 1 flips bit.

From K&R: Write a function, invert(x,p,n), that returns x with the n bits that begin at position p inverted, leaving the others unchanged.

```c
unsigned int invert(unsigned int x, unsigned int p, unsigned int n){
    unsigned int constant = 1;
    int i = 0;
    for (i = 1; i < n; i++){
        constant |= constant << 1;
    }
    constant = constant << p;
    return x ^ constant;
}
```

Part II: C and Pointers

Write the following functions so that they perform according to the following descriptions.

1. Write a program that prints the string "Hello World" to standard output. For this question only, write the entire contents of the C file and try to make your solution stylistically ideal.
```c
#include <stdio.h>
#include <stdlib.h>

void hello_world();

int main(int argc, char *argv[]) {
    hello_world();
    return EXIT_SUCCESS;
}

void hello_world() {
    printf("Hello World
");
}

2. Swaps the value of two ints outside of this function.

void swap(int* a, int* b){
    int temp = *a;
    *a = *b;
    *b = temp;
}

3. Increments the value of an int outside of this function by one.

void increment(int *x) {
    (*x)++;
}

/* OR */
void increment(int *x) {
    x[0]++;
}

4. Returns the number of bytes in a string. Does not use strlen

int mystrlen(char* str) {
    int count = 0;
    while (*str++) {
        count++;
    }
    return count;
}

5. Returns the number of elements in an array ARR of ints. The array must be able to store any integer that fits in the int type.

You can’t. C has no way to determine an end of a sequence of ints.