CS61C Fall 2013 – 2 – Practice With C

Pointers

1. What is terrible about the following function?

```
int *blasphemy(void) {
        int x;
        return &x;
}
```

This function returns a pointer to memory on the stack, but deallocated after the function returns. There's no guarantee that this memory will contain or hold any particular value, or that things won't crash when you try to access it.

2. What is wrong about the following function that swaps the values of two int variables? What changes need to be made for swappy to function correctly?

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

Arguments are passed by value in C, so we would be operating on *copies* of the ints we intended to swap. Thus the original function wouldn't actually do anything.

3. What does the following function do?

```
int mystery(int *arr, int n) {
      return n ? arr[0] + mystery(arr + 1, n -1) : 0;
}
```

Returns the sum of the first n elements of the array arr.

Bitwise Operators

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.

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

x & 0 = _0__   x | 0 = x___   x ^ 0 = _x__

x & 1 = _x_   x | 1 = 1__   x ^ 1 = ~x_

2. 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

CS61C Fall 2013 – 2 – Practice With C

C Programming Practice

Complete the implementation of the following functions based on the comments.

1. Increments the value of an int outside this function by one.
```
void increment( int *x ) {
      (*x)++;                        //or x[0]++;
}
```

2. Returns the number of bytes within a string. Do not use strlen().
```
int mystrlen( char* str ) {
      int count = 0;
      while (*str++) {
            count++;
      }
      return count;
}
```

3. Returns the number of elements in an array *arr of ints. The array must be able to store any integer that fits in the array.

   You can't. C has no way to determine the length of an array.

Structs

Structs are user-defined collections of variables. A structure definition goes as follows:
```
struct structure_tag {
      type1 member1;
      ...
      typen membern;
};
```
What does each of the four following statements do?

`struct {int x; int y;} var;`

Defines a variable `var` of an unnamed structure type that contains two integers `x` and `y`.

`struct point {int x; int y;};`

Defines the type `struct point`.

`struct point {int x; int y;} pt1;`

Defines the variable `pt1` of type `struct point`.

`struct point {int x; int y;} pt1 = {1,2};`

Defines the variable `pt1` of type `struct point` and initializes it to `pt1.x=1` and `pt2.y=2`.