Week 16 - Introductory Programming Lab

Exercise 14: Bit manipulation

Background

As engineers, you will (or *should* at least!) have more interest in bitwise manipulation than most other programmers. Writing software that controls lights, reacts to flipped switches, and generally anything that interacts closely with hardware will require bitwise operations. The main concepts here are *flags* and *bitmasks* (commonly known as "masks).

A *flag* is a particular bit with a special meaning. For example, when representing negative numbers, the 8th bit could be thought of a flag which indicated whether the remainder of the bits should be interpreted as a positive or negative number.

A *bitmask* is a way of accessing a particular bit (generally, but not always, a flag). The bitmask is a number which has 0 in all bits we don't care about, and a 1 for the bit(s) that we want to examine. By *and*ing the bitmask with the original number, we can "extract" the bit(s) -- if that bit was 0, then the new number will be completely zero; if the bit was 1, then the new number will be non-zero.

The same operation is done in reverse to set a flag -- by *or*ing a bitmask and the data variable, we can set a flag to be true. Setting a flag can be done by *and*ing the variable with the *not*ed bitmask.

Technical details

Various bitwise operations:

```
#include <stdio.h>

int main() {
    char x = 0x10; // number in hex
    char y = 63;

    printf("char is a one-byte number.  ");
    printf("Printing it with %%c will treat ");
    printf("it as\nan ASCII value, but we ");
    printf("can also print it as a number: ");
    printf("x=%i y=%i.\n", x, y);

    printf("Some operations: ");
    printf("%i\t%i\n", ~x, ~y);
    printf("%i\t%i\t%i\n", x&y, x|y, x^y);

    getchar();
    return 0;
}
```

Toggling individual bits:

```
#include <stdio.h>
#include <stdlib.h>  // extra includes!
#include <time.h>

float getRand() {
    return rand() / (RAND_MAX+1.0);
}

int main() {
    srand( time(NULL) ); // init random
    char x = 0;
    char bit, bitmask;
    int i;

    // Building a bitwise random number
    for (i=0; i<8; i++) {
        // set bit as 1 or 0?
        if (getRand() > 0.5) {
            bit = 1;
        } else {
            bit = 0;
        }
        // get bit ready
        bitmask = bit << i;
        // or them together
        x = x | bitmask;
        printf("After round %i, ",i);
        printf("bit is %i, and ", bit);
        printf("x is %i.\n", x);
    }

    getchar();
    return 0;
}
```

Your task...

Dougie Dog has invented an "encryption" technique to keep secrets from the Ferocious Kittens. Fortunately, cats are extremely intelligent, and have cracked the simple code:

1. Letters are grouped into pairs. Add a space to the end of the string if necessary to give it an even number of characters (here "character" means `char`.
2. Make an `int` from each pair by sticking the bits from the first letter in front of the bits from the second letter. You may assume that we are using 8-bit ASCII.
3. XOR the result with 31337.

   Here's two examples of encryption: "cats" and "kittens".

   1. Pairs: "ca ts"
      "ki tt en s_" (_ represents a space)

2. into ints: 25441 29811
   27497 29812 25966 29472
3. XOR with 31337: 6408 3610
   4352 3613 7943 2377

Decryption is performed with the same steps, but in reverse order.

- The Ferocious Kittens have intercepted two secret messages from Dougie Dog:

  ```
  15643 6913 6916 23040 2377 6985 6408 3657 5638 3084 2119
  15910 23079 13629 23101 10300 10557 23073 13092 23369
  ```

- Write a program that decrypts them.
  (hint: this will be a lot easier if you begin by writing a program to *encrypt* values -- you can check each step with "cats" and "kittens" to make sure you understand the process!)
- You *must* use a function to split a large integer into two separate letters. This function *may not* print anything to the screen.
  (hint: how can a function return two values?)

Show your work to a demonstrator/GTA.

Exercise 15: Pointers

Background

Pointers: stuff from lecture notes.

To keep track of time, computers need something to measure it against. In C on the lab computers, we get the number of seconds since 00:00:00 UTC on 1 January 1970.

As you might imagine, this produces a fairly large number -- it exceeds the bounds of a normal `int`. We therefore use a `long int`, which has twice the number of bits of a normal `int`. Our `time.h` library also defines a special data type to store the time value: `time_t`. On our computers, this is the same as a `long int`.

Technical details

Swapping data with pointers:

```
#include <stdio.h>

void swap(int *px, int *py) {
    int temp = *px;
    *px = *py;
    *py = temp;
}

int main() {
    int x = 1;
    int y = 2;
    printf("initial: %i %i\n", x, y);
```

```
    swap( &x, &y);
    printf("swapped: %i %i\n", x, y);

    getchar();
    return 0;
}
```

Watch out for the * and &

Getting the time:

```
#include <stdio.h>
#include <time.h>  // extra include

int main() {
    time_t now;
    now = time(NULL);

    long int a = now;
    printf("%li\n", now); }

    getchar();
    return 0;
}
```

Your task...

Calling `time(NULL)` gives you a large integer. Use this value to calculate today's year, date, hours, and minutes.

- Simplifying assumption: we pretend that each year has exactly 365 days. The 15th of March is therefore day 83 in year 2011. (use that date to figure out today's "day number")
  (hint: how many seconds are there in a day?)
- You *cannot* use `struct` for this assignment.
- You *cannot* use any time-based functions from the standard library such `asasctime` or `strftime`.
- You *must* use one function to calculate the year and day (pass in the total seconds, and a reference to the year and day).
- You *must* use a separate function to calculate the hours and minutes. (again, pass the total seconds, and a reference to the hours and minutes)
- You *must* use a separate function to print the time; this function can only take as arguments the year, day, hour, and minutes.

(optional: handle leap-years correctly, and print the month as well)

Show your work to a demonstrator/GTA.