Week Two

VideoPlus

The following comes from http://www.newscientist.com :

''The VideoPlus system works quite simply by using a reversible algorithm, something well known to most computer programmers. These types of algorithms are found in diverse applications, ranging from credit card machines through to data compression and encryption.

The VideoPlus algorithm itself is a closely guarded secret, but it's probably only a little more complicated than the example described below. I just wish I'd had the marketing nous to think of it first.

A simple example of such an algorithm would work along the following lines. Imagine you wish to record a programme that starts at 9.50 pm on 3 November. VideoPlus probably uses a cyclic date system, rather than the standard way of representing dates. If, for example, there are 50 days in a VideoPlus "month", then it would be possible to use VideoPlus up to 50 days in advance of a programme.

So, 3 November might be converted to, say, day 45. You would then convert 9.50 pm to the 24-hour clock and append the VideoPlus day to the converted time. This gives 215045. The next step would be to work out by what single-digit number 215045 could be divided by without leaving a fraction. Five is an obvious answer, so let's use that. Divide 215045 by five and you get 43009. Append the dividing factor to the end of that number and you have a "VideoPlus" number of 430095. Now, unless you know the algorithm, it won't be easy to convert that number into a time and date.''

- Write a function that tells you what "VideoPlus" day it is today, in the range `0 - 49`. To help with this use the `time()` function which returns the number of seconds since the 1/1/1970:

```
#include <sys/types.h>
#include <time.h>
#include <stdio.h>

int main(void)
{
   printf("There have been %ld seconds since 1/1/1970\n", time(NULL));
   return 0;
}
```

- Write a program that takes in the time of the program to be recorded, the number of days into the future that it is and produces a "VideoPlus" number as described above. Reuse the function written above to help with this.

Bitwise Dice

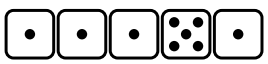
Many dice games, for example *Yahtzee* :

`http://www.hasbro.com/common/instruct/Yahtzee.pdf`

involve throwing five dice, then allowing the player to roll some (or all of the dice again).
For example, on the first throw you might get :

The player might want to keep the ones and re-roll the other dice. They might get :

and for their third and final throw (choosing to only roll two of the dice) they might get :



The object of this assignment is primarily about how to specify which of the dice to throw again. We will achieve this using a binary representation. There are five dice. Therefore a 5-bit binary number can specify which of the dice to re-roll ("1") or which to leave alone ("0"). For instance, in the first example above, we wanted to re-roll the middle three dice ($01110_2 = 14_{10}$). In the next case we wanted to re-roll the third and fourth dice ($00110_2 = 6_{10}$).
Write a program that shows the users first (random) throw, and then allows them to re-roll another two times. An example my look like :

```
$ diceroll
You got :
1 4 6 2 1
Which dice to roll again ? 14
You got :
1 1 3 5 1
Which dice to roll again ? 6
You got :
1 1 1 5 1
$
```

Hints

You can check whether a particular bit of a number is set or not by using the bitwise "and" in C. This is via the `&` operator. So, to check if the third bit is set, the C code would look something like :

```
if((i & 4) > 0){
    printf("It's set\n");
}
else{
    printf("It's not set\n");
}
```

To check the fourth bit, you'd use 8 in the above code rather than 4 etc.

A good tutorial on binary and bitwise operators in C can be found by clicking the following :

http://www.gamedev.net/page/resources/_/technical/general-programming/bitwise-operations-in-c-r1563