CS 7A - Spring 2015 - Converting between base 2, base 10 and base 26. Due 2/10/15

Background Theory

To protect your identity when your grades are posted to the internet, I've added a level of encryption that converts your student number (base 10) and converts it to base 26 where we use $A = 0, B = 1, C = 2, \ldots, Z = 25$ are the multipliers of powers of 26.

Let's start with the method of converting from base 10 to base 2. It turns out to be fairly easy to generalize this algorithm to other bases.
Here is pseudocode for accomplishing the conversion:

```
number = positive integer

while (number > 0 )
{
  bit    = number mod 2 ;
  number = number div 2 ;
  put bit to the left of any previous bits
}
```

The reason this works is that the base 2 where the ith digit is d_i version of the number can written in this form:

$$N = d_0 + d_1 \cdot 2 + d_2 \cdot 2^2 + \cdots + d_n \cdot 2^n$$

and the ith division by 2 and produces the remainder d_i in turn.
For instance, $237 = 1 + 0 \cdot 2 + 1 \cdot 4 + 1 \cdot 8 + 0 \cdot 16 + 1 \cdot 32 + 1 \cdot 64 + 1 \cdot 128 = 11101101_2$
In class we started by writing this code

```
1  /** Geoff Hagopian
   Converting from base 10 to base 26 and using A=0,B=1,...,Z=25 as place values. **/
3
   #include "std_lib_facilities.h"
5
   //------------------------------------------------------------------------------
7
   int main() {
9      int x;
       cout << "\nEnter␣and␣integer␣to␣convert␣to␣base␣2\n";
11     cin >> x;
       cout << x%2; //the remainder after division by 2
13     x /= 2; //
       cout << x%2;
15     x /= 2;
       cout << x%2;
17     x /= 2;
       //...keep repeating this until it's done...how do we know?
19 }
   //------------------------------------------------------------------------------
```

After working a number of these conversions by hand and taking note of the what happens to `x` when it's repeatedly divided by 2 (discarding the remainder at each step) we observe that we are done when the integer part of half of x is zero. This suggests using a `while` loop and condition the loop on `x` not being zero:

```
//beginning comments and preprocessor commands omitted for brevity

int main() {
    int x;
    cout << "\nEnter␣and␣integer␣to␣convert␣to␣base␣2\n";
    cin >> x;
    while(x!=0) { // note that "!" is the negation operator.
        cout << x%2; //the remainder after division by 2
            x /= 2; // integer division discards remainder, so 3/2 is 1.
        }
}
//-------------------------------------------------------------------------
```

Entering 237 here yields the following

```
Enter and integer to convert to base 2
237
10110111
```

This is fine except the the digits are in reverse order. What we did in class to fix this is to store the digits as characters in a `string`. To do this we had to learn about casting an integer as a character. The ASCII character `'0'` has the integer value 48, so `char(48);` will print `'0'` will 'cast' the integer 48 as the `char`acter `'0'`. Starting with an empty string, `binary`, then we can use concatenation to append either a `'1'` or a `'0'` with the command `binary = char(48+x%2)+binary`, which adds to the left side of the string at each iteration:

```
int main() {
    int x;
    cout << "\nEnter␣and␣integer␣to␣convert␣to␣base␣2\n";
    cin >> x;
    while(x!=0) { // note that "!" is the negation operator.
        binary = char(48+x%2)+binary; // can't use += here, why not?
        x = x/2;
        }
}
//-------------------------------------------------------------------------
```

1. Adapt this code to convert from base 10 to base 26 and use this to convert your student number to a base 26 "number." Check that 18048_{10} = BASE.