Password Program

Input length of password from command line. Ex 3.
Find all possible sets of 3 digits. There are 120 possible sets of 3 digits.

Ex: 012
013
014
015
…
689
789

Take one set of 3 digits. Find all ways of rearranging (i.e. permuting) those digits. There are 6 possible ways to rearrange 3 digits:

Ex. 012
021
102
120
201
210

After finding each permutation, send it to pass.o to check if it's the password.

There are a total of 720 potential passwords of length 3.

How to find the sets:

This is where using binary comes in.

Given a binary number, count the number of ones. You want it to match the set size (in this case 3).

If it matches, convert from a binary number to set of digits. The concept here is similar to the Sieve of Eratosthenes that was covered at the start of class: A 1 in the binary number means the index of that spot is in the set.

Binary	0	**1**	0	0	**1**	0	**1**	0	0	0
Index	0	**1**	2	3	**4**	5	**6**	7	8	9

The set would be: 1, 4, 6

Binary	0	0	**1**	**1**	0	0	0	0	0	**1**
Index	0	1	**2**	**3**	4	5	6	7	8	**9**

The set would be: 2, 3, 9

What your code may look like:

In main:

```
- input password length from command line (N)
- loop from i =0 to i = 2^9 (this loops from 0000000000 to 1111111111)
        - find the binary version of i. Try saving it in an array. (can modify baser.cpp)
        - count the number of 1s in the binary
        - if the number of 1s is the same as N
                - convert to a set (will need to be stored in an array)
                - find permutations of the set
                        (use the permutation.cpp code. will probably need to modify it)
                - try each permutation with pass.o
                - if you find the password, print it out
```

pass.o accepts a character array. You can convert each password you want to try from an int array to a char array using a modified itos. Instead of converting to a string, convert to a c-string (char array). Or use char arrays for the whole program.