Interview Question

- Write a function that enumerates all possible subsets of a given string.
    - Example:
        - Given string is “ab”
            - Your function generates:
            - “” , “a”, “b”, “ab”
        - Given string is “abc”
            - Your function generates:
            - “” , “a”, “b”, “c”, “ab”, “bc”, “ac”, “abc”

- There are many ways to solve this.

- One trick that u can use to enumerate all possible subsets is to use changes in the bit pattern in an integer representation, as you change the integer value.

- Example:
  - int i;
  - For (i = 0; i < 8; ++ ii)

← Bit pattern in integer ii binary representation changes with change in its value.

How can u use that?

- Here's what I mean:
  - For this explanation, just assume we have an int that is 3 bits long.
  - 3 bits can represent values from 0 to 7 (so, a total of 8 values, i.e., 2^3 values)
  - These will be:
    - 000
    - 001
    - 010
    - 011
    - 100
    - 101
    - 110
    - 111

- Now, if u r given the string "abc" and u have to enumerate all its subsets, how do we use the above binary bit pattern?

- We can think of it as:
  - each letter of the string corresponds to a bit in the binary representation:

- a b c
- ----- ----- -----
- bit 2 bit 1 bit 0

- Again, the subset enumeration we need to do will generate this:
  - “” , “a”, “b”, “c”, “ab”, “bc”, “ac”, “abc”

- Do u see the algorithm now?

Desired output: “” , “c”, “a”, “b”, “bc”, “ac”, “ab”, “abc”

Lets take the first one, which is the empty set: “”

- “” would correspond to bit pattern for 0 which is 000, indicating **none** of the letters are present.
- And “c” would correspond to bit pattern for 1 which is 001, indicating letter associated with the last bit is present, which is letter ‘c’
- “b” would correspond to bit pattern for 2 which is 010
- “bc” would correspond to bit pattern for 3 which is 011
- And so on.

- a b c
- ----- ----- -----
- bit 2 bit 1 bit 0

Q: How many subsets will a string of length N have ?

So, the pseudo code would look something like:

```
List_of_strings  EnumerateSubsets( string str )
{
        List_of_strings results;
        int length = str.length();
        ulong numSubsets = power(2, length);

        for (int ii = 0; ii < numSubsets; ++ ii)
        {
                string subsetStr = GenerateOneSubsetFromBitPattern( ii, str );
                results.Add (subsetStr );
        }

        return results;
}
```

```
string GenerateOneSubsetFromBitPattern( int value, strring inputStr )
{
        int numberOfBitsInInt = sizeof( int ) * 8; //  sizeof returns size of int in bytes, multiply by 8 to get number of bits
        int numBitsToCheck = Min( numberOfBitsInInt, inputStr.Length() );
        string subsetStr;

        for ( int i = 0; i < numBitsToCheck; ++ i )
        {
                // Check if the ith bit of value is 1:
                // We do this by  left shifting 0x1 (which is bit pattern 01 ) by i places.
                // This puts the 1 in 0x1 at the   ith position.
                // And when we do a bit and with value, it tells us if the  ith bit in value is 1 or 0

                int bitPatternWithIthBitAs1 =   0x1 << i ;        // Left shift 01 by i positions.
                int ithBitValue = value  &  bitPatternWithIthBitAs1;

                if ( ithBitValue  != 0 )
                        subsetStr = subsetStr + inputStr[ i ];       // Take ith bit of str
        }
        return subsetStr;
}
```

Limitations of this algorithm?