Write a computer program in the language of your choice which reads in a single non-negative integer N as input and produces as output the **power set** of {1, 2, 3, ..., N}, written one subset per output line. For example, if N = 4, your output should look something like:

{ 1 2 3 4 }
{ 1 2 3 }
{ 1 2 4 }
{ 1 2 }
{ 1 3 4 }
{ 1 3 }
{ 1 4 }
{ 1 }
{ 2 3 4 }
{ 2 3 }
{ 2 4 }
{ 2 }
{ 3 4 }
{ 3 }
{ 4 }
{ }

The order of the listing is unimportant; the example above suggests one possible natural ordering but there are other natural orderings too.

Hand in a hard-copy of your source code, and hard-copy of the output your program produces for N = 5.

Programming ideas / suggestions:

(1). There are at least two approaches to solving this problem, one recursive and one iterative. Feel free to use either method.
(2). The recursive method will require you to use some sort of data structure (preferably a *stack*) to keep track of which "subset" a particular level of the recursion is working on. (Of course, this stack is in addition to the computer's own stack which your recursive program "automatically" utilizes, compliments of how the compiler translates your code!) The stack that you the programmer must implement is, as usual, most easily done via an array or a linked list. If you opt for an array, you can assume N <= 20 for array declaration purposes.
(3). The iterative version may in fact be conceptually easier. We merely picture the numbers 0 through 2^N -1 written as N-digit binary numbers, and think of the binary digit 1 as meaning "select" and the binary digit 0 as meaning "don't select". The iterative version can be written in such a manner that it requires no data structures at all to program.