CSE143 Section #13 Problems

1. Write a recursive method called printNumbers that prints all of the integers that are composed of two 2's and two 5's. Your method should write each combination on a separate line of output.

2. Write a recursive method called solve that takes integers x and y as parameters and that prints all solutions for traveling from (0, 0) to (x, y) using one of three moves:

```
North (abbreviation N):      move up 1 (increase y)
East (abbreviation E):       move right 1 (increase x)
NorthEast (abbreviation NE): move up 1 and right 1 (increase both)
```

   For example, if passed the values 1 and 2, it should print the following solutions (the solutions can appear in any order):

```
solutions:
moves: N N E
moves: N E N
moves: N NE
moves: E N N
moves: NE N
```

3. Write a recursive method called printNumbers2 that prints all of the integers that are composed of one 1, two 2's, and five 3's. Your method should write each combination on a separate line of output.

4. Write a recursive method called printSubsets that takes an array of integer values and that prints all subsets of the list. For example, if we have:

```
int[] list = {1, 2, 3};
```

   and we make the call:

```
printSubsets(list);
```

   The method should print the 8 subsets:

```
[]
[3]
[2]
[2, 3]
[1]
[1, 3]
[1, 2]
[1, 2, 3]
```

   Your method can print the subsets in any order, but you should preserve the order of the values from the list. For example, the subsets of [42, 23] should be listed as:

```
[]
[23]
[42]
[42, 23]
```

   You may assume the list has no duplicate values.

5. Write a recursive method called printBinary that takes an integer n as a parameter and that counts in binary using n digits, printing each value on a separate line. All n digits should be shown for all numbers. For example, if n is 2, the output should be:

```
00
01
10
11
```

When n is equal to 3, the output should be:

```
000
001
010
011
100
101
110
111
```

Your method should throw an IllegalArgumentException if n is less than 0 and should produce no output if n is equal to 0.

6. Write a recursive method called partitionable that takes a List<Integer> as a parameter and that returns true if the list can be partitioned into two sublists of equal sum and that returns false otherwise. The table below indicates various possible values for a variable list and the value that would be returned by the call list.partitionable():

List Contents	Value Returned	List Contents	Value Returned
[]	true	[2, 1, 8, 3]	false
[42]	false	[8, 8]	true
[1, 2, 3]	true	[-3, 14, 3, 8]	true
[1, 2, 3, 4, 6]	true	[-4, 5, 7, 2, 9]	false

For example, the list [1, 2, 3] can be split into [1, 2] and [3], both of which have a sum of 3. The list [1, 2, 3, 4, 6] can be split into [1, 3, 4] and [2, 6], both of which have a sum of 8. For the list [2, 1, 8, 3], there is no way to split the list into two sublists whose sum is equal.

You may assume that you can add and remove at the front of the list in O(1) time, so you are allowed to modify the list as you compute the answer as long as you restore it to its original form.