Solution to CSE143 Section #13 Problems

1. Two possible solutions appear below (only the private method is different):

```
public static void printNumbers() {
    explore(2, 2, "");
}

private static void explore(int twos, int fives, String number) {
    if (twos == 0 && fives == 0) {
        System.out.println(number);
    } else if (twos >= 0 && fives >= 0) {
        explore(twos - 1, fives, number + "2");
        explore(twos, fives - 1, number + "5");
    }
}
```

```
private static void explore(int twos, int fives, String number) {
    if (twos == 0 && fives == 0) {
        System.out.println(number);
    } else {
        if (twos > 0) {
            explore(twos - 1, fives, number + "2");
        }
        if (fives > 0) {
            explore(twos, fives - 1, number + "5");
        }
    }
}
```

2. One possible solution appears below.

```
public void solve(int x, int y) {
    System.out.println("solutions:");
    explore(0, 0, x, y, "moves:");
}

private void explore(int currX, int currY, int x, int y, String path) {
    if (currX == x && currY == y) {
        System.out.println(path);
    } else if (currX <= x && currY <= y) {
        explore(currX, currY + 1, x, y, path + " N");
        explore(currX + 1, currY + 1, x, y, path + " NE");
        explore(currX + 1, currY, x, y, path + " E");
    }
}
```

3. One possible solution appears below.

```
public static void printNumbers2() {
    explore(1, 2, 5, "");
}
```

```
    private static void explore(int ones, int twos, int threes,
                                String number) {
        if (ones == 0 && twos == 0 && threes == 0) {
            System.out.println(number);
        } else if (ones >= 0 && twos >= 0 && threes >= 0) {
            explore(ones - 1, twos, threes, number + "1");
            explore(ones, twos - 1, threes, number + "2");
            explore(ones, twos, threes - 1, number + "3");
        }
    }
```

4. One possible solution appears below.

```
    public void printSubsets(int[] list) {
        Stack<Integer> subset = new Stack<>();
        explore(subset, list, 0);
    }

    private void explore(Stack<Integer> subset, int[] list, int index) {
        if (index >= list.length) {
            System.out.println(subset);
        } else {
            explore(subset, list, index + 1);
            subset.push(list[index]);
            explore(subset, list, index + 1);
            subset.pop();
        }
    }
```

5. One possible solution appears below.

```
    public void printBinary(int n) {
        if (n < 0) {
            throw new IllegalArgumentException();
        }
        if (n > 0) {
            print(n, "");
        }
    }

    private void print(int digitsLeft, String s) {
        if (digitsLeft == 0) {
            System.out.println(s);
        } else {
            print(digitsLeft - 1, s + "0");
            print(digitsLeft - 1, s + "1");
        }
    }
```

6. One possible solution appears below.

```
  private boolean explore(List<Integer> rest, int sum1, int sum2) {
```

```
    if (rest.isEmpty()) {
        return sum1 == sum2;
    } else {
        int n = rest.remove(0);
        boolean result = explore(rest, sum1 + n, sum2) ||
                         explore(rest, sum1, sum2 + n);
        rest.add(0, n);
        return result;
    }
}

public boolean partitionable(List<Integer> list) {
    return explore(list, 0, 0);
}
```