CSCE 340 PROJECT 4 BACKTRACKING ALGORITHMS

DR. GODFREY MUGANDA

1. Dividing an Inheritance

A person who owns n items with integer dollar values $v_0, \ldots v_{n-1}$ wishes to have the items divided among two children for an inheritance, in such a way that the two children each end up with collections of equal total worth. For example, if there are three items worth 3, 10, and 7, then one child gets the \$10 item while the other child gets the \$3 and \$7 items.

Clearly, a fair and equitable division is not always possible, as for example when there are only 2 items worth \$12 and \$10. In this case, the instructions in the will are to give the estate to charity.

Your job is to write a backtracking solution that reads in an array of input values and presents a fair split of the items when possible, or states that a fair division is not possible.

There will be a single file containing multiple data sets for this problem. The format of the data will be as follows. The file starts off with an integer N, the number of data sets in the file, followed by N data sets. Each data set starts of with an integer m, the number of items in the data set, followed by m integers, which are the dollar values of the m items.

Here is an example of a sample file.

```
3
4 1 5 11 5
6
3
1 1 2 2 1
3
5 14 8
```

2. Sample Input/Output

Here is a sample run on this data set:

```
Select a file that contains inheritance data
Original input is [1, 5, 11, 5]
Original sorted input is [1, 5, 5, 11]
A fair split is [1, 5, 5] and [11].
Original input is [3, 1, 1, 2, 2, 1]
Original sorted input is [1, 1, 1, 2, 2, 3]
A fair split is [1, 1, 3] and [1, 2, 2].
Original input is [5, 14, 8]
Original sorted input is [5, 8, 14]
A Fair split is not possible, Donate estate to charity.
```

Note that your backtracking solution must work on the *unsorted* input: the printing of the sorted version is to aid in checking and debugging.

3. Code Organization

Structure your code in two files as follows.

(1) `EstateSplit`: this is a class with static methods that implement the backtracking strategy. This class will have a public static method

```
boolean solve(int [] inputs, LinkedList<Integer> part1,
              LinkedList<Integer> part2
             )
```

that takes an array of input values to be split, and two empty lists `part1` and `part2`. If the split is successful, the two lists `part1` and `part2` will be filled with the values of the items in such a way that each item value is included in exactly one of the parts and the two parts have equal sum, and the `solve()` method returns true. However, if a fair split is not possible, then the `solve()` method returns false.

Naturally, there will be other methods in the `EstateSplit` class, including an `extend()` method that performs the backtracking part.

(2) `EstateSplitMain`: This is a driver file that reads the input from the files and calls the `solve()` method. This code will be given to you.

4. Due Date

Due March 13 2020