COMP 610 Project 3: Brute Force LoadBalancing

Points: 30 points possible

Overview: You will be given an instance of LoadBalancing and determine the optimal solution to this instance.

Details: The input will come from a file called input.txt which will be placed in the same directory as your java file. The first line of the file will have a single integer value N which will be the number of items. The remaining lines will be N whitespace separated values (the items).

You will determine the optimal solution for this instance. In other words you will find the smallest possible value M for which the items can be split into 2 sets where both sets have the total of items no larger tham M. Since LoadBalancing is known to be NP-complete. Your algorithm will likely need to utilize brute force.

You can discuss ideas for the algorithm to be used with anyone and consult any source (books, internet, etc). However, for this project, you are expected to write the code on your own with limited or no assistance from the professor, no assistance from others, and limited or no assistance from other sources (books, internet, etc). To clarify, you can seek assistance in understanding the task and how it can be solved, but "your code" should be written by you: not written by others, not copied from others, not copied from books/internet.

Picky, but required specifications: Your project must:

- be submitted via canvas.
- consist of 1 or more dot-java files (no class files, zip files, input files or other files should be submitted). Each file must have your name and which project you are submitting as comments on the first 2 lines.
- not be placed into any package (for the java pedants, it must be in the default package).
- have one file called Project3.java.
- compile using the command 'javac Project3.java'.
- run using the command 'java Project3',
- accept input from a file called input.txt in the same directory as the java file(s) formatted precisely as described above.
- be submitted on time (early and multiple times is fine).

If your project fails any of the above, you will receive a zero (recall that each of you may replace 1 and only 1 project that receives a zero this semester). If your project meets the requirements above then it will be graded on whether it:

- is designed and formatted reasonably (correct indentation, no excessively long lines, no excessively long methods, has useful method/variable names, etc) and
- accomplishes the goal of the project. In other words, the output should be the correct answer, computed in a valid way, formatted correctly.

Sample execution: If input.txt contains

```
5
2
2
3
3
3
```

then the output should be

```
7
```

because you could split the items up $\{2, 2, 3\}$ and $\{3, 3\}$.

If input.txt contains

```
6
5
7
9
6
8
5
```

then the output should be

```
20
```

because you could split the items up $\{5, 9, 6\}$ and $\{7, 8, 5\}$.

If input.txt contains

```
5
7
6
5
19
5
```

then the output should be

```
23
```

because you could split the items up $\{19\}$ and $\{7, 6, 5, 5\}$.

Stray Thoughts:
I suggest you finish and submit your project at least several days in advance. This way you have time and opportunity to ask any last questions and verify that what you upload satisfies the requirements. There is nothing wrong with working on the project for a day and uploading your code, working for another day and uploading your improved code, ..., working for another day and uploading your final version. In fact there are advantages: you have a fairly reliable place that keeps your versions and even if you get busy at the last moment you have still uploaded your best version.
Your project should be written and understood by you. Helping or receiving help from others to figure out what is allowed/required is fine, but copying code is not. Significant shared source code indicates that you either did not write/understand what you submitted or you provided code to another (allowing them to submit code they did not write/understand). Both are academic dishonesty.