Problem Instructions:

1. This is a timed exercise; however, it is on the honor system. We expect you to spend no more than two hours total over the next 48 hours to complete it. No need to rush. We don’t consider how long it takes you to complete the problem so please take the time to check your work thoroughly before submission.

2. You must work independently. You cannot consult with anyone else.

3. If you do not finish within the time allotted, please submit what you have completed and provide an explanation in your email of what work remains to be done. If you have problems with your development environment, please notify us immediately via email by responding to this message.

4. We will accept solutions in Java, Python. Please use only standard libraries and write your code to be portable. You are free to use your favorite IDE, editor, and reference documentation.

5. Your solution will be judged primarily for its correctness. However, we also place significant emphasis on code clarity, design and efficiency.

6. If you have questions about the problem, please use your best judgment and carefully document your assumptions in the code.

7. When you have your final solution, please reply to this email with your solution as a .zip, .tar, or .tar.gz attachment. Do not include any binaries, only source code.

Project title: Optimal Partition

Given a list of integer items, we want to partition the items into at most N non-overlapping, consecutive bins, in a way that minimizes the maximum number of items in any bin.

For example, suppose we are given the items (5, 2, 3, 6, 1, 6), and we want 3 bins. We can optimally partition these as follows:

```
n < 3:      1, 2 (2 items)
3 <= n < 6: 3, 5 (2 items)
6 <= n:     6, 6 (2 items)
```

Every bin has 2 items, so we can’t do any better than that.

Your task: write a program to partition a set of input items. It should be runnable as

```
<yourprogram | java -jar yourprogram | python yourprogram> <input> <output>
```

Input:

A text file, with one integer per line. The first line is the maximum number of bins, N. Subsequent lines are the items to be binned (some of which may be duplicates).

Output:

A text file with one bin per line. For each bin, print a space-separated list of the items in that bins. The bins and items should be in ascending order.

Example

Example input:

```
3
5
2
3
6
1
6
```

Example output:

```
1 2
3 5
6 6
```

Your solution will be evaluated on the size of the partitions it generates (how close to optimal they are); its computational efficiency (time and space requirements), and by the correctness, readability, and quality of the code.