Scooters relocation

Input	inputX.txt
Output	outputX.txt

Imagine a city with n scooters in areas with low demand. The same city has m parking spaces in areas with high demand. We'd like to move as many scooters as possible from where they are to those parking spaces (no more than one scooter per parking space).
The scooters will be transported using k vans. The van fleet is all starting off at a single point. Each one will then follow a route that picks scooters up and drops them off at parking spaces in a set order. But there are some constraints for the routes.

- Each van has a maximum route length.
- Each van can fit 25 scooters. That means that **at any given time**, they can hold no more than 25 scooters. Of course, the overall route can move more than 25 scooters.

Working within those restrictions, we'd like to move as many scooters as possible from where they are to those parking spaces (no more than one scooter per parking space).
For simplicity, we'll assume that there are $n + m + 1$ locations:

- One starting point with k vans (point with index 0);
- n scooters (starting points with indexes $1, \ldots, n$);
- m parking spaces (points with indexes $n + 1, \ldots, n + m$).

The distance between the points is given by square matrix D, where the element $D(i, j)$ sets the distance from point i to point j ($0 \le i, j \le n + m$).
Triangle inequality is valid for the distance matrix: for any three points v_1, v_2, v_3, the inequality $D(v_1, v_3) \le D(v_1, v_2) + D(v_2, v_3)$ holds. It therefore doesn't make sense for vans to make intermediate stops, i.e., visiting a point without picking up or dropping off a scooter there.
The solution will be a set of k routes (possibly empty) that **do not visit the same points** while satisfying the route length and van capacity requirements.

Input format

The input data is in files *input1.txt*, *input2.txt*, …, *input30.txt*. Download the archive from the task page and extract it.
The first line of the input file contains the positive integers n, m, and k ($1 \le n, m \le 1000$, $2 \le k \le 5$), which are the number of scooters, parking spaces, and vans, respectively.
The following $n + m + 1$ lines each contain $n + m + 1$ numbers — the distance matrix D ($0 \le D(i, j) \le 10^5$, $0 \le i, j \le n + m$). The i-th ($0 \le i \le n + m$) line has the elements $D(i, 0)$, $D(i, 1)$, …, $D(i, n + m)$.

In other words, the matrix is given like this:

$$\begin{matrix} \mathrm{D}(0,0) & \mathrm{D}(0,1) & \dots & \mathrm{D}(0,\mathrm{n}+\mathrm{m}) \\ \mathrm{D}(1,0) & \mathrm{D}(1,1) & \dots & \mathrm{D}(1,\mathrm{n}+\mathrm{m}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{D}(\mathrm{n}+\mathrm{m},0) & \mathrm{D}(\mathrm{n}+\mathrm{m},1) & \dots & \mathrm{D}(\mathrm{n}+\mathrm{m},\mathrm{n}+\mathrm{m}) \end{matrix}$$

It is guaranteed that there are zeros on the main diagonal of the matrix. But the matrix itself may not be symmetric.
The last line of the input file contains k positive numbers d_1, d_2, ..., d_k ($0 \le d_i \le 10^5$) — restrictions on van route lengths.

Output format

Your job is to send us a **zip** archive with files *output1.txt*, *output2.txt*, ..., *output30.txt*, where output file *outputX.txt* matches input file *inputX.txt*. If any of the files are missing from your archive, the test will be ignored.
The output file should consist of k lines. The i-th line should contain the route of the i-th van. The first thing in the i-th line should be a number s_i — the number of points on the i-th van's route not counting the starting point (with index 0). Separated by a space in the same line, you should have s_i indexes for the stops on the route.
Make sure you add all your source code to the zip file you send. It will **not** be launched on the server, though we will make sure it's there at the end of the competition.
Evaluation

We'll first check to make sure your routes follow the rules.

- They should not intersect at any stops or contain point 0.
- The route length for the i-th van should not exceed d_i. Route length is the sum of the distances between consecutive points in the route. The distance from starting point 0 to the first stop on the route is also included.
- No van can have more than 25 scooters at any point.
- **Vans should not have any scooters left in them at the end of their route.**
- When a van stops at a spot with a scooter, it must have less than 25 scooters, and it must have at least one when stopping by a parking space.

If the output set of routes meets the requirements, you will score one point for each scooter transported. Otherwise, your solution will earn zero points.

Your total score will be the sum of your individual test scores.
500 points for this problem equals one qualifying point. Thus, in order to qualify for the finals of the Algorithm direction, you must score at least 10 000 points.

Notes

Example

```
Input:
3 4 2
0 1 2 4 3 4 3 1
1 0 1 5 2 3 4 2
2 1 0 6 1 2 5 3
4 5 6 0 7 8 1 3
3 2 1 7 0 1 6 4
4 3 2 8 1 0 7 5
3 4 5 1 6 7 0 2
1 2 3 3 4 5 2 0
5 4
Output:
2 3 6
4 1 2 4 5
```

This solution is optimal since all three scooters were transported. The first van's route length is 5, while the second van's is 4.

In the above example, the distance matrix is symmetric, but this is not necessarily true for the rest of the tests.