**Disk Defragment**

**Problem**

When files are stored on a hard disk, they often become fragmented. This means that the file is not stored in sequential sectors on the disk. The first half of a file might be stored in sector 243, while the second half of a file might be far away in sector 105.

The goal of defragmenting a hard drive is to arrange the files so that each file is stored in order on sequential sectors of the disk. Thus, if a file required 4 sectors of storage space, it would end up in sectors N, N+1, N+2, and N+3, for some N.

Typically, this is done to increase the overall speed of the computer.

There are a number of programs that will defrag a hard disk as described above. However, many of them are painfully slow. You are trying to develop a new algorithm to defrag hard drives, but before you start, you would like to determine how fast you can defrag a very small drive without very many files on it.

You will be given the locations of a number of files on a small hard disk, and are to determine the minimum number of sectors that must be moved before the entire drive is defragged. You have enough memory to hold two sectors worth of data at once, but that is all.

You will be given a ‘string[] disk' each of whose elements represents a single file.

Each element of disk will be formatted as a single-space delimited list of integers which represent the locations of the parts of the file, in order.

Hence, the String, "4 9 6 59 41" represents a file stored in 5 sectors where the first part of the file is in sector 4 of the disk.

One way to defrag this file would be to move the contents of sector 9 to sector 5, the contents of sector 59 to sector 7, and the contents of sector 41 to sector 8. By doing this, the file would be stored sequentially in sectors 4-8. You will also be given an ‘int size' representing the total number of sectors on the disk (sectors 0 through size-1, inclusive, may contain data).

You are to return the smallest number of sectors that must be moved to defragment the whole disk.

Keep in mind that you can not move data to a sector until any data being stored there is moved.

**Definition**

Class : DiskDefrag

Method : MinMoves

Parameters : string[], int

Returns : int

Method signature : public int MinMoves(string[] disk, int size)

**Constraints**

* 10 ≤ size ≤ 100 [between 10 and 100, inclusive]
* Disk will contain between 1 and 12 elements, inclusive.
* Each element of disk will contain between 1 and 50 characters, inclusive.
* Each element of disk will be a single-space delimited list of integers, without extraneous leading zeros.
* Each integer in disk will be between 0 and size-1, inclusive.
* No integer will be appear more than once in disk.

**Sample**

3 4 5 6 8 9 10

17 16 15

20

Returns: 5

We can defrag the first file by moving the contents of sector 8 to sector 7, then 9 to 8, and finally 10 to 9.

The second file can be defragged in a number of ways by moving the contents of two sectors, for a total of 5.

1 2 3 5 4 6 7 8

10

Returns: 2

Here we can take advantage of the fact that we have enough memory to hold two sectors worth of data.

First, load the contents of sectors 4 and 5 into memory. Now, simply write the data back in the reverse order.

1 3 5 7

0 2 4 8

6 9

100

Returns: 7