Great Computer Challenge

Scientific/Non-Business Programming

Level IV

Introductory comments; please read carefully!

1. All the problems below require an input file with a specific name. The name is given in the description of the corresponding problem. You are advised not to change the filename since this may confuse your judge – which is clearly not what you want!

2. Your team must specify the directory for each program where the judges can put that input file. If no location is given, the judges will assume the same directory as the program's executable.

3. If your team anticipates compatibility problems, for example with non-Windows or Linux systems, tell the judges how to create a proper input file. The same holds for copying a file to the desired location.

4. As usual you can assume a certain level of tolerance and imagination with the judges **BUT** if your output varies from the expected pattern described in the problem, you are advised to explain how to read your program's results.

Combination Lock

The combination lock of this problem, as shown in the figure, consists of a circular dial, which can be turned (clockwise or counterclockwise) and is embedded into the "fixed" part of the lock. The dial has *N* evenly spaced "ticks". The ticks are numbered from 0 to *N*-1, increasing in the clockwise direction.
The fixed part of the lock has a red “mark" which always "points to" a particular tick on the dial. Of course, the mark points to different ticks as the dial is turned. (In the figure, $N = 40$ and the mark points to tick 14.)

The lock comes with three code numbers $T1$,$T2$, $T3$. These are non-negative integers and each of them is less than *N*. No two of the three are the same.

The lock is opened in three stages of operations:

1. If the mark initially points to tick $T1$, turn the dial exactly two full revolutions clockwise and stop. Otherwise, turn the dial clockwise exactly two full revolutions clockwise and continue to turn it clockwise until the mark points to tick $T1$.
2. Turn the dial one full revolution counterclockwise and continue to turn it counterclockwise until the mark points to tick $T2$.
3. Turn the dial clockwise until the mark points to tick $T3$. The lock should now open.

Given the numbers *N*, $T1$, $T2$, $T3$, the goal of this problem is to find the maximum number of ticks the dial must be turned in order to open the lock according to the three stages of operations outlined above. The program reads in a file called “combination.txt”. The file contains several lines where each line holds a set of numbers in the order given above and separated by a single space. The input list is terminated by a line containing a single ‘0’, e.g.

40 7 25 13
80 63 42 9
60 12 53 33
0

Lottery

Fred likes to play the lotto. Whenever he does, he buys lots of tickets. Each ticket has 6 unique numbers in the range from 1 to 30, inclusive. Fred likes to "Cover all his bases." By that, he means that he likes for each set of lottery tickets to contain every number from 1 to 30, at least once, on some ticket. Write a program to help Fred see of his tickets "Cover all the bases."

Input

The input file is called "lottery.txt". It consists of a number of test cases. Each case starts with an integer N (between 1 and 100, inclusive), indicating the number of tickets Fred has purchased. On the next N lines are the tickets, one per line. Each ticket will have exactly 6 integers, and all of them will be in the range from 1 to 30 inclusive. No ticket will have duplicate numbers, but the numbers on a ticket may appear in any order. The input ends with a line containing only a '0'.

Output

Print a list of responses for the input sets, one per line. Print the word 'Yes' if every number from 1 to 30 inclusive appears in some lottery ticket in the set, and `No' otherwise. Print these words exactly as they are shown. Do not print any blank lines between outputs.

Sample Input (lottery.txt)

```
1
1 2 3 4 5 6
5
1 2 3 4 5 6
10 9 8 7 12 11
13 14 15 16 17 18
19 20 21 22 23 24
25 26 27 28 29 30
6
1 2 3 4 5 6
10 9 8 7 12 11
13 14 15 16 17 18
13 20 25 22 27 24
25 26 27 28 29 30
10 20 18 17 12 11
0
```

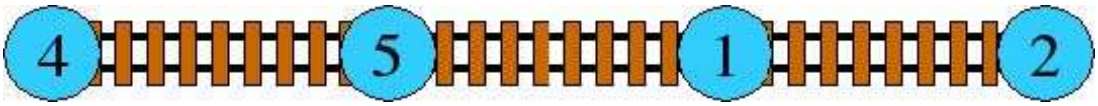
Sample Output

```
No
Yes
No
```

Lawrence of Arabia

T. E. Lawrence was a controversial figure during World War I. He was a British officer who served in the Arabian theater and led a group of Arab nationals in guerilla strikes against the Ottoman Empire. His primary targets were the railroads. A highly fictionalized version of his exploits was presented in the blockbuster movie, "Lawrence of Arabia".

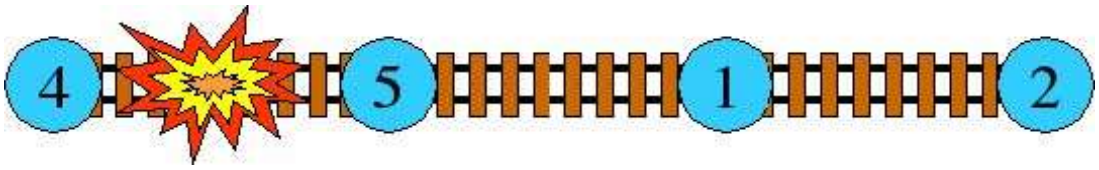
You are to write a program to help Lawrence figure out how to best use his limited resources. You have some information from British Intelligence. First, the rail line is completely linear---there are no branches, no spurs. Next, British Intelligence has assigned a Strategic Importance to each depot---an integer from 1 to 5. A depot is of no use on its own, it only has value if it is connected to other depots. The Strategic Value of the entire railroad is calculated by adding up the products of the Strategic Values for every pair of depots that are connected, directly or indirectly, by the rail line. Consider this railroad:



Its Strategic Value is 4*5 + 4*1 + 4*2 + 5*1 + 5*2 + 1*2 = 49.

Now, suppose that Lawrence only has enough resources for one attack. He cannot attack the depots themselves---they are too well defended. He must attack the rail line between depots, in the middle of the desert. Consider what would happen if Lawrence attacked this rail line right in the middle:

The Strategic Value of the remaining railroad is 4*5 + 1*2 = 22. But, suppose Lawrence attacks between the 4 and 5 depots:



The Strategic Value of the remaining railroad is 5*1 + 5*2 + 1*2 = 17. This is Lawrence's best option.

Given a description of a railroad, figure out the smallest Strategic Value that Lawrence can achieve for that railroad with one attack.

Input

The input file is called "loa.txt". It contains several data sets. Each data set will begin with a line with one integer *n* the number of depots on the railroad ($1 \le n \le 20$). On the next line will be *n* integers, each from 1 to 5, indicating the Strategic Value of each depot in order. End of input will be marked by a line with $n=0$, which should not be processed.

Output

For each data set, output a single integer, indicating the smallest Strategic Value for the railroad that Lawrence can achieve with his attack. Output each integer in its own line.

Sample Input (loa.txt)

```
4
4512
5
45123
0
```

Sample Output

```
17
31
```

Pathfinder Problem

Imagine a chessboard like grid with 10 rows and 10 columns, making up one hundred cells. Some of these cells are "impassable". Your program reads in some unknown number of impassable cells from a file called "obstacles.txt". For each impassible cell x,y coordinates are listed in the file as shown below. Each coordinate in the pair is an integer between 0 and 9. The list is terminated by a row with the coordinates 0 0.

Your program should then display the grid using X to denote impassible cells and C to denote clear ones.

Finally your program should determine whether a path exists from the upper LEFT hand corner of the grid (0,0) to the lower RIGHT hand corner (9,9). The only allowed moves from one cell to another are left or right and up or down; diagonal moves are not permitted. Your program should display a message saying if a path exists or not.

A sample data file with the appropriate grid display is shown below, as well as the correct run time response.

obstacles.txt
```
3 3
3 4
3 5
6 2
9 7
8 8
8 9
0 0
```

Grid
```
C  C  C  C  C  C  C  C  C  C
C  C  C  C  C  C  C  C  C  C
C  C  C  C  C  C  C  C  C  C
C  C  C  X  X  X  C  C  C  C
C  C  C  C  C  C  C  C  C  C
C  C  C  C  C  C  C  C  C  C
C  C  X  C  C  C  C  C  C  C
C  C  C  C  C  C  C  C  C  C
C  C  C  C  C  C  C  C  X  X
C  C  C  C  C  C  C  X  C  C
```

Solution: No path exists.