[All Tracks](https://www.hackerearth.com/practice/) **🡪**[Data Structures](https://www.hackerearth.com/practice/data-structures/)**🡪**[Arrays](https://www.hackerearth.com/practice/data-structures/arrays/)**🡪**[Multi-dimensional](https://www.hackerearth.com/practice/data-structures/arrays/multi-dimensional/)**🡪**

**Roy and Symmetric Logos**

Roy likes Symmetric Logos.

How to check whether a logo is symmetric?  
Align the center of logo with the origin of Cartesian plane. Now if the colored pixels of the logo are symmetric about both X-axis and Y-axis, then the logo is symmetric.

You are given a binary matrix of size **N x N** which represents the pixels of a logo.  
**1** indicates that the pixel is colored and **0** indicates no color.

For instance: Take a 5x5 matrix as follows:

01110

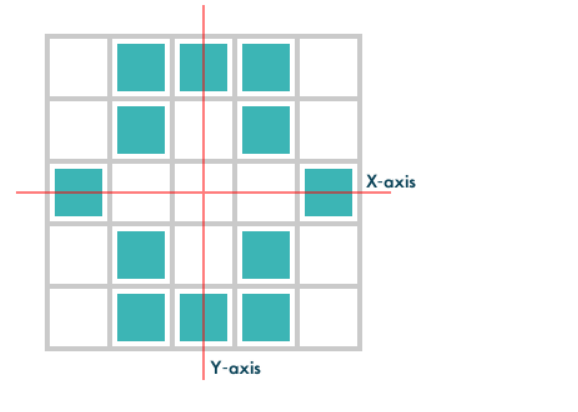
01010

10001

01010

01110

Graphically it is represented as follows:



Observe that it is symmetric about both X-axis and Y-axis.

Let's take another example of 5x5 matrix:

00100

01010

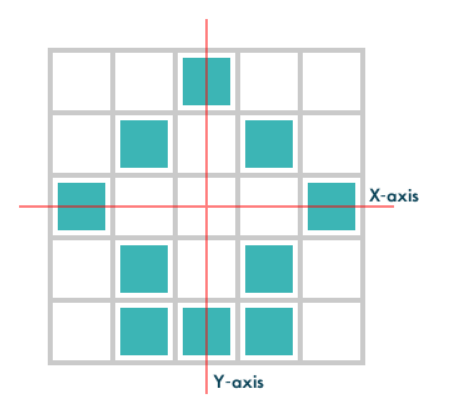
10001

01010

01110

Graphically it is represented as follows:

Now this logo is symmetric about Y-axis but it is **not symmetric** about X-axis.



Your task is to output YES if the logo is symmetric else output NO.

Input:  
First line contains T - number of test cases.  
T test cases follow.  
First line of each test case contains the N - size of matrix.  
Next N lines contains binary strings of length N.

Output:  
Print YES or NO in a new line for each test case

Constraints:   
1 ≤ T ≤ 10  
2 ≤ N ≤ 32

Note: There will always be at least 1 colored pixel in input data.

**SAMPLE INPUT**

5

2

11

11

4

0101

0110

0110

0101

4

1001

0000

0000

1001

5

01110

01010

10001

01010

01110

5

00100

01010

10001

01010

01110

**SAMPLE OUTPUT**

YES

NO

YES

YES

NO

/\* My C Solution \*/

#include <stdio.h>

#define N 33

int main()

{

int arr[N][N],n,t,i,j,k,xflag,yflag;

scanf("%d",&t);

for(i=0;i<t;i++){

scanf("%d",&n);

for(j=0;j<n;j++)

for(k=0;k<n;k++)

scanf("%1d",&arr[j][k]);

xflag=yflag=0;

for(j=0;j<n;j++){

for(k=0;k<n;k++){

if(arr[j][k]==arr[n-1-j][k])

xflag=1;

else{

xflag=0;

break;

}

if(arr[j][k]==arr[j][n-1-k])

yflag=1;

else{

yflag=0;

break;

}

}

if(k!=(n/2))

if((!xflag && yflag) || (xflag && !yflag))

break;

}

if(xflag && yflag)

printf("YES\n");

else

printf("NO\n");

}

return 0;

}

/\*Editorial

**Author Solution** by [Ravi Ojha](https://www.hackerearth.com/@akatsuki)

1. def check\_about\_x\_axis(n, matrix):
2. for i in xrange(n):
3. j = n-i-1
4. for p in xrange(n):
5. if matrix[i][p] != matrix[j][p]:
6. return 0
7. return 1
8. def check\_about\_y\_axis(n, matrix):
9. for i in xrange(n):
10. j = n-i-1
11. for p in xrange(n):
12. if matrix[p][i] != matrix[p][j]:
13. return 0
14. return 1
15. def roy\_and\_symmetric\_logos():
16. t = input()
17. for tt in xrange(t):
18. n = input()
19. matrix = []
20. for i in xrange(n):
21. s = raw\_input()
22. tmp = list(s)
23. matrix.append(tmp)
24. if check\_about\_y\_axis(n,matrix) and check\_about\_x\_axis(n,matrix):
25. print "YES"
26. else:
27. print "NO"
28. roy\_and\_symmetric\_logos()

\*/

Input #1:

9

2

11

11

2

01

00

2

10

00

2

00

10

2

00

01

4

0101

0110

0110

0101

4

1001

0000

0000

1001

5

01110

01010

10001

01010

01110

5

00100

01010

10001

01010

01110

Output #1:

YES

NO

NO

NO

NO

NO

YES

YES

NO

Input #2:

10

2

11

00

2

00

11

2

10

01

2

01

10

2

11

11

2

10

10

2

11

10

2

11

01

2

01

11

2

10

11

Output #2:

NO

NO

NO

NO

YES

NO

NO

NO

NO

NO

Input #3:

10

3

111

000

111

3

111

001

111

3

111

010

111

3

111

111

111

3

111

101

111

3

111

101

101

3

110

101

011

3

010

101

010

3

110

101

011

3

000

101

000

Output #3:

YES

NO

YES

YES

YES

NO

NO

YES

NO

YES

Input #4:

10

3

110

101

011

3

010

000

010

4

1101

0110

0110

1101

4

1001

0110

0110

1001

4

1001

0111

0110

1001

4

1001

0100

0110

1001

4

1000

0000

0000

1001

4

1000

0000

0000

0001

4

1001

0000

0000

0001

4

0000

0000

0010

0000

Output #4:

NO

YES

NO

YES

NO

NO

NO

NO

NO

NO

Input #5:

8

32

00000000000000011000000000000000

00000000000000100100000000000000

00000000000001000010000000000000

00000000000010000001000000000000

00000000000100000000100000000000

00000000001000000000010000000000

00000000010000000000001000000000

00000000100000000000000100000000

00000001000000000000000010000000

00000010000000000000000001000000

00000100000000000000000000100000

00001000000000000000000000010000

00010000000000000000000000001000

00100000000000000000000000000100

01000000000000000000000000000010

10000000000000000000000000000001

10000000000000000000000000000001

01000000000000000000000000000010

00100000000000000000000000000100

00010000000000000000000000001000

00001000000000000000000000010000

00000100000000000000000000100000

00000010000000000000000001000000

00000001000000000000000010000000

00000000100000000000000100000000

00000000010000000000001000000000

00000000001000000000010000000000

00000000000100000000100000000000

00000000000010000001000000000000

00000000000001000010000000000000

00000000000000100100000000000000

00000000000000011000000000000000

32

00000000000000011000000000000000

00000000000000100100000000000000

00000000000001000010000000000000

00000000000010000001000000000000

00000000000100000000100000000000

00000000001000000000010000000000

00000000010000000000001000000000

00000000100000000000000100000000

00000001000000000000000010000000

00000010000000000000000001000000

00000100000000000000000000100000

00001000000000000000000000010000

00010000000000000000000000001000

00100000000000000000000000000100

01000000000000000000000000000010

10000000000000011000000000000001

10000000000000000000000000000001

01000000000000000000000000000010

00100000000000000000000000000100

00010000000000000000000000001000

00001000000000000000000000010000

00000100000000000000000000100000

00000010000000000000000001000000

00000001000000000000000010000000

00000000100000000000000100000000

00000000010000000000001000000000

00000000001000000000010000000000

00000000000100000000100000000000

00000000000010000001000000000000

00000000000001000010000000000000

00000000000000100100000000000000

00000000000000011000000000000000

32

00000000000000011000000000000000

00000000000000100100000000000000

00000000000001000010000000000000

00000000000010000001000000000000

00000000000100000000100000000000

00000000001000000000010000000000

00000000010000000000001000000000

00000000100000000000000100000000

00000001000000000000000010000000

00000010000000000000000001000000

00000100000000000000000000100000

00001000000000000000000000010000

00010000000000000000000000001000

00100000000000000000000000000100

01000000000000000000000000000010

00000000000000011000000000000000

00000000000000011000000000000000

01000000000000000000000000000010

00100000000000000000000000000100

00010000000000000000000000001000

00001000000000000000000000010000

00000100000000000000000000100000

00000010000000000000000001000000

00000001000000000000000010000000

00000000100000000000000100000000

00000000010000000000001000000000

00000000001000000000010000000000

00000000000100000000100000000000

00000000000010000001000000000000

00000000000001000010000000000000

00000000000000100100000000000000

00000000000000011000000000000000

32

00000000000000011000000000000000

00000000000000100100000000000000

00000000000001000010000000000000

00000000000010000001000000000000

00000000000100000000100000000000

00000000001000000000010000000000

00000000010000000000001000000000

00000000100000000000000100000000

00000001000000000000000010000000

00000010000000000000000001000000

00000100000000000000000000100000

00001000000000000000000000010000

00010000000000000000000000001000

00100000000000000000000000000100

01000000000000000000000000000010

10000000000000011000000000000001

00000000000000011000000000000000

01000000000000000000000000000010

00100000000000000000000000000100

00010000000000000000000000001000

00001000000000000000000000010000

00000100000000000000000000100000

00000010000000000000000001000000

00000001000000000000000010000000

00000000100000000000000100000000

00000000010000000000001000000000

00000000001000000000010000000000

00000000000100000000100000000000

00000000000010000001000000000000

00000000000001000010000000000000

00000000000000100100000000000000

00000000000000011000000000000000

31

0000000000000011000000000000000

0000000000000100100000000000000

0000000000001000010000000000000

0000000000010000001000000000000

0000000000100000000100000000000

0000000001000000000010000000000

0000000010000000000001000000000

0000000100000000000000100000000

0000001000000000000000010000000

0000010000000000000000001000000

0000100000000000000000000100000

0001000000000000000000000010000

0010000000000000000000000001000

0100000000000000000000000000100

1000000000000000000000000000010

1000000000000000000000000000001

0100000000000000000000000000010

0010000000000000000000000000100

0001000000000000000000000001000

0000100000000000000000000010000

0000010000000000000000000100000

0000001000000000000000001000000

0000000100000000000000010000000

0000000010000000000000100000000

0000000001000000000001000000000

0000000000100000000010000000000

0000000000010000000100000000000

0000000000001000001000000000000

0000000000000100010000000000000

0000000000000010100000000000000

0000000000000001000000000000000

31

0000000000000001000000000000000

0000000000000010100000000000000

0000000000000100010000000000000

0000000000001000001000000000000

0000000000010000000100000000000

0000000000100000000010000000000

0000000001000000000001000000000

0000000010000000000000100000000

0000000100000000000000010000000

0000001000000000000000001000000

0000010000000000000000000100000

0000100000000000000000000010000

0001000000000000000000000001000

0010000000000000000000000000100

0100000000000000000000000000010

1000000000000000000000000000001

0100000000000000000000000000010

0010000000000000000000000000100

0001000000000000000000000001000

0000100000000000000000000010000

0000010000000000000000000100000

0000001000000000000000001000000

0000000100000000000000010000000

0000000010000000000000100000000

0000000001000000000001000000000

0000000000100000000010000000000

0000000000010000000100000000000

0000000000001000001000000000000

0000000000000100010000000000000

0000000000000010100000000000000

0000000000000001000000000000000

31

0000000000000001000000000000000

0000000000000010100000000000000

0000000000000100010000000000000

0000000000001000001000000000000

0000000000010000000100000000000

0000000000100000000010000000000

0000000001000000000001000000000

0000000010000000000000100000000

0000000100000000000000010000000

0000001000000000000000001000000

0000010000000000000000000100000

0000100000000000000000000010000

0001000000000000000000000001000

0010000000000000000000000000100

0100000000000000000000000000010

0000000000000000000000000000000

0100000000000000000000000000010

0010000000000000000000000000100

0001000000000000000000000001000

0000100000000000000000000010000

0000010000000000000000000100000

0000001000000000000000001000000

0000000100000000000000010000000

0000000010000000000000100000000

0000000001000000000001000000000

0000000000100000000010000000000

0000000000010000000100000000000

0000000000001000001000000000000

0000000000000100010000000000000

0000000000000010100000000000000

0000000000000001000000000000000

31

1000000000000000000000000000001

0000000000000010100000000000000

0000000000000100010000000000000

0000000000001000001000000000000

0000000000010000000100000000000

0000000000100000000010000000000

0000000001000000000001000000000

0000000010000000000000100000000

0000000100000000000000010000000

0000001000000000000000001000000

0000010000000000000000000100000

0000100000000000000000000010000

0001000000000000000000000001000

0010000000000000000000000000100

0100000000000001000000000000010

1000000000000000000000000000001

0100000000000001000000000000010

0010000000000000000000000000100

0001000000000000000000000001000

0000100000000000000000000010000

0000010000000000000000000100000

0000001000000000000000001000000

0000000100000000000000010000000

0000000010000000000000100000000

0000000001000000000001000000000

0000000000100000000010000000000

0000000000010000000100000000000

0000000000001000001000000000000

0000000000000100010000000000000

0000000000000010100000000000000

1000000000000000000000000000001

Output #5:

YES

NO

YES

NO

NO

YES

YES

YES

Input #6:

5

2

10

01

4

1001

0110

0110

1001

4

1001

0000

0000

1001

5

01110

01010

10101

01010

01110

5

00100

01010

10001

01010

01110

Output #6:

NO

YES

YES

YES

NO

Input #7:

10

3

000

010

000

4

1001

1001

1001

1001

4

1111

0000

0000

1111

5

00100

01010

00100

01010

00100

5

00100

01010

10101

01010

00100

5

00100

00000

00100

00000

00100

5

00000

00000

11111

00000

00000

5

10000

01000

00100

00010

00001

5

10001

00010

00100

01000

10001

6

001100

001100

111111

001100

111111

001100

Output #7:

YES

YES

YES

YES

YES

YES

YES

NO

NO

NO

Input #8:

10

6

000000

010010

000000

000000

010010

000000

8

10011001

10011001

10011001

11111111

11111111

10011001

10011001

10011001

8

11111111

00001111

00001111

11111111

11111111

00001111

00001111

11111111

10

0010000100

0101001010

0010000100

0101001010

0010000100

0010000100

0101001010

0010000100

0101001010

0010000100

10

0010000100

0101001010

1010110101

0101001010

0010000100

0010000100

0101001010

1010110101

0101001010

0010000100

10

0000000000

0100000010

0000000000

0000000000

0000000000

0000000000

0000000000

0000000000

0100000010

0000000000

10

0000000000

0100000010

0000000000

0000000000

0000000001

0000000001

0000000000

0000000000

0100000010

0000000000

10

1000000001

0100000010

0000000000

0000000000

1111111111

0000000000

0000000000

0000000000

0100000010

1000000001

10

0000000000

0111111110

0000000000

0000110000

0000110000

0000110000

0000110000

0000000000

0111111110

0000000000

12

001100001100

111111111111

001100001100

111111111111

001100001100

001100001100

001100001100

001100001100

111111111111

001100001100

111111111111

001100001100

Output #8:

YES

YES

NO

YES

YES

YES

NO

NO

YES

YES

Input #9:

10

32

00000000000000000000000000000000

11111111111111111111111111111111

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000010000000000001000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

11111111111111111111111111111111

11111111111111111111111111111111

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000010000000000001000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

11111111111111111111111111111111

00000000000000000000000000000000

32

10000000000000000000000000000001

01000000000000000000000000000010

00100000000000000000000000000100

00010000000000000000000000001000

00001000000000000000000000010000

00000100000000000000000000100000

00000010000000000000000001000000

00000001000000000000000010000000

00000000100000000000000100000000

00000000010000000000001000000000

00000000001000000000010000000000

00000000000100000000100000000000

00000000000010000001000000000000

00000000000001000010000000000000

00000000000000100100000000000000

00000000000000011000000000000000

11111111111111111111111111111111

00000000000000100100000000000000

00000000000001000010000000000000

00000000000010000001000000000000

00000000000100000000100000000000

00000000001000000000010000000000

00000000010000000000001000000000

00000000100000000000000100000000

00000001000000000000000010000000

00000010000000000000000001000000

00000100000000000000000000100000

00001000000000000000000000010000

00010000000000000000000000001000

00100000000000000000000000000100

01000000000000000000000000000010

10000000000000000000000000000001

32

10000000000000011000000000000001

01000000000000011000000000000010

00100000000000011000000000000100

00010000000000011000000000001000

00001000000000111100000000010000

00000100000000111100000000100000

00000010000000011000000001000000

00000001000000011000000010000000

00000000100000011000000100000000

00000000010000011000001000000000

00000000001000011000010000000000

00000000000100011000100000000000

00000000000010011001000000000000

00000000000001111110000000000000

00000000000000100100000000000000

10000000000000011000000000000001

11111111111111111111111111111111

00000000000000100100000000000000

00000000000001111110000000000000

00000000000010011001000000000000

00000000000100011000100000000000

00000000001000011000010000000000

00000000010000011000001000000000

00000000100000011000000100000000

00000001000000011000000010000000

00000010000000011000000001000000

00000100000000111100000000100000

00001000000000111100000000010000

00010000000000011000000000001000

00100000000000011000000000000100

01000000000000011000000000000010

10000000000000011000000000000001

32

10000000000000011000000000000001

01000000000000011000000000000010

00100000000000011000000000000100

00010000000000011000000000001000

00001000000000111100000000010000

00000100000000111100000000100000

00000010000000011000000001000000

00000001000000011000000010000000

00000000100000011000000100000000

00000000010000011000001000000000

00000000001000011000010000000000

00000000000100011000100000000000

00000000000010011001000000000000

00000000000001111110000000000000

00000000000000100100000000000000

11111111111111111111111111111111

11111111111111111111111111111111

00000000000000100100000000000000

00000000000001111110000000000000

00000000000010011001000000000000

00000000000100011000100000000000

00000000001000011000010000000000

00000000010000011000001000000000

00000000100000011000000100000000

00000001000000011000000010000000

00000010000000011000000001000000

00000100000000111100000000100000

00001000000000111100000000010000

00010000000000011000000000001000

00100000000000011000000000000100

01000000000000011000000000000010

10000000000000011000000000000001

10

1000000001

0100000010

0000000000

0000000000

1011111111

1111111101

0000000000

0000000000

0100000010

1000000001

20

00111111000011111100

00111111000011111100

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001111111111110000

00001111111111110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00111111000011111100

00111111000011111100

32

00000000000000000000000000000000

11111111111111111111111111111111

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000010000000000001000000000

00000000010000000000001000000000

00000000010000000000001000000000

00000000010000000000001000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000010000000000001000000000

00000000000000000000000000000000

00000000000000000000000000000000

11111111111111111111111111111111

11111111111111111111111111111111

00000000000000000000000000000000

00000000000000000000000000000000

00000000010000000000001000000000

00000000000000000000000000000000

00000000010000000000001000000000

00000000010000000000001000000000

00000000010000000000001000000000

00000000010000000000001000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

11111111111111111111111111111111

00000000000000000000000000000000

24

001100001100001100001100

111111111111111111111111

001100001100001100001100

111111111111111111111111

001100001100001100001100

001100001100001100001100

001100001100001100001100

001100001100001100001100

111111111111111111111111

001100001100001100001100

111111111111111111111111

001100001100001100001100

001100001100001100001100

111111111111111111111111

001100001100001100001100

111111111111111111111111

001100001100001100001100

001100001100001100001100

001100001100001100001100

001100001100001100001100

111111111111111111111111

001100001100001100001100

111111111111111111111111

001100001100001100001100

10

1000000001

0100000010

0000000000

0000110000

1011001101

1011001101

0000110000

0000000000

0100000010

1000000001

20

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001111111111110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

Output #9:

YES

NO

NO

YES

NO

YES

NO

YES

YES

NO

Input #10:

10

32

00000000000000000000000000000000

00000000000000000000000000000000

00111111100000000000000111111100

00111111100000000000000111111100

00001110000000000000000001110000

00001110000000000000000001110000

00001110000000000000000001110000

00001110000000000000000001110000

00001110000000000000000001110000

00001110000000000000000001110000

00001110000000000000000001110000

00001110000000000000000001110000

00001110000000000000000001110000

00001110000000000000000001110000

00001110000000000000000001110000

00001111111111111111111111110000

00001111111111111111111111110000

00001110000000000000000001110000

00001110000000000000000001110000

00001110000000000000000001110000

00001110000000000000000001110000

00001110000000000000000001110000

00001110000000000000000001110000

00001110000000000000000001110000

00001110000000000000000001110000

00001110000000000000000001110000

00001110000000000000000001110000

00001110000000000000000001110000

00111111100000000000000111111100

00111111100000000000000111111100

00000000000000000000000000000000

00000000000000000000000000000000

32

00000000000000000000000000000000

00000000000000000000000000000000

00001111111111111111111111110000

00001111111111111111111111110000

00001110000000000000000001110000

00001110000000000000000001110000

00001110000000000000000000000000

00001110000000000000000000000000

00001110000000000000000000000000

00001110000000000000000000000000

00001110000000000000000000000000

00001110000000000000000000000000

00001110000000000000000000000000

00001110000000000000000000000000

00001110000000000000011100000000

00001111111111111111111100000000

00001111111111111111111100000000

00001110000000000000011100000000

00001110000000000000000000000000

00001110000000000000000000000000

00001110000000000000000000000000

00001110000000000000000000000000

00001110000000000000000000000000

00001110000000000000000000000000

00001110000000000000000000000000

00001110000000000000000000000000

00001110000000000000000001110000

00001110000000000000000001110000

00001111111111111111111111110000

00001111111111111111111111110000

00000000000000000000000000000000

00000000000000000000000000000000

32

10000000000000011000000000000001

01000000000000011000000000000010

00100000000000011000000000000100

00100000000000000000000000000100

00100000000000000000000000000100

00000100000000111100000000100000

00000010000000011000000001000000

00000001000000011000000010000000

00000000100000011000000100000000

00000000010000011000001000000000

00000000001000011000010000000000

00000000000100011000100000000000

00000000000010011001000000000000

00000000000001000010000000000000

00000000000000100100000000000000

10000000000000011000000000000001

10000000000000011000000000000001

00000000000000100100000000000000

00000000000001000010000000000000

00000000000100011000100000000000

00000000001000011000010000000000

00000000010000011000001000000000

00000000100000011000000100000000

00000001000000011000000010000000

00000010000000011000000001000000

00000100000000111100000000100000

00100000000000000000000000000100

00100000000000000000000000000100

00100000000000000000000000000100

00100000000000011000000000000100

01000000000000011000000000000010

10000000000000011000000000000001

10

1000000001

0100000010

1011111111

0000000000

0000000000

0000000000

0000000000

1011111111

0100000010

1000000001

20

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00111111000011111100

00111111000011111100

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

31

0000000000000000000000000000000

1111111111111111111111111111111

0010000000000000000000000000100

0010000000000000000000000000100

0010000000000000000000000000100

0010000000000000000000000000100

0010000000000000000000000000100

0010000000000000000000000000100

0010000000000000000000000000100

0010000000000000000000000000100

0010000000000000000000000000100

0010000000000000000000000000100

0010000000000000000000000000100

0000000000000100010000000000000

0000000000000010100000000000000

1000000000000001000000000000001

0000000000000010100000000000000

0000000000000100010000000000000

0010000000000000000000000000100

0010000000000000000000000000100

0010000000000000000000000000100

0010000000000000000000000000100

0010000000000000000000000000100

0010000000000000000000000000100

0010000000000000000000000000100

0010000000000000000000000000100

0010000000000000000000000000100

0010000000000000000000000000100

0010000000000000000000000000100

1111111111111111111111111111111

0000000000000000000000000000000

24

001100001100001100001100

111111111111111111111111

001100001100001100001100

111111111111111111111111

001100001100001100001100

001100001100001100001100

001100001100001100001100

001100001100001100001100

111111111111111111111111

111111111111111111111111

111111111111111111111111

111111111111111111111111

111111111111111111111111

111111111111111111111111

111111111111111111111111

111111111111111111111111

001100001100001100001100

001100001100001100001100

001100001100001100001100

001100001100001100001100

111111111111111111111111

001100001100001100001100

111111111111111111111111

001100001100001100001100

10

1000000001

0100000010

0000000000

0000110000

1011001101

1011001101

0000110000

0000000000

0100000010

1000000001

20

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001111111111110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

00001100000000110000

32

10000000000000000000000000000001

01000000000000000000000000000010

00111111111111111111111111111100

00100000000000000000000000000100

00100000000000000000000000000100

00100000000000000000000000000100

00100000000000000000000000000100

00100000000000000000000000000100

00100000000000000000000000000100

00100000000000000000000000000100

00100000000000000000000000000100

00100000000000000000000000000100

00100000000000000000000000000100

00100000000000000000000000000100

00100000000000000000000000000100

00100000000000000000000000000100

00100000000000000000000000000100

00100000000000000000000000000100

00100000000000000000000000000100

00100000000000000000000000000100

00100000000000000000000000000100

00100000000000000000000000000100

00100000000000000000000000000100

00100000000000000000000000000100

00100000000000000000000000000100

00100000000000000000000000000100

00100000000000000000000000000100

00100000000000000000000000000100

00100000000000000000000000000100

00111111111111111111111111111100

01000000000000000000000000000010

10000000000000000000000000000001

Output #10:

YES

NO

NO

NO

YES

YES

YES

YES

NO

YES

/\* Best Submissions:

C:

1. *#include*<stdio.h>
2. *#include*<stdlib.h>
3. char p[32][32];
4. int n;
5. int main(void)
6. {
7. int i,j,t,m,s;
8. scanf("%d",&t);
9. while(t>=1)
10. {
11. scanf("%d",&n);
12. getchar();
13. for(i=0;i<n;i++)
14. {
15. for(j=0;j<n;j++)
16. scanf("%c",&p[i][j]);
17. getchar();
18. }
19. m=n/2-1;
20. s=0;
21. for(i=0;i<=m;i++)
22. {
23. for(j=0;j<n;j++)
24. if(p[i][j]==p[n-1-i][j]&&p[j][i]==p[j][n-1-i])
25. {
26. s++;
27. }
28. }
29. if(s==n\*(n/2))
30. printf("YES\n");
31. else
32. printf("NO\n");
33. t--;
34. }
35. }

C++:

1. *#include* <bits/stdc++.h>
2. using namespace std;
4. int main(){
5. int t,f,i,j,n;
6. cin>>t;
7. while(t--){
8. cin>>n;
9. char a[n][n];
10. for(i=0;i<n;i++)
11. cin>>a[i];
12. for(i=0;i<n;i++){
13. for(j=0;j<n;j++){
14. if(a[i][j] == a[n-i-1][j] && a[i][j]==a[i][n-j-1]){
15. f=1;
16. continue;
17. }
18. else{
19. f=0;break;
20. }
21. }
22. if(f==0)
23. break;
24. }
25. if(f)
26. cout<<"YES"<<endl;
27. else
28. cout<<"NO"<<endl;
29. }
30. return 0;
31. }

C++14:

1. *#include*<bits/stdc++.h>
2. using namespace std;
4. int main()
5. {
6. int t;
7. cin>>t;
8. while(t--)
9. {
10. int n;
11. cin>>n;
12. string s[n];
14. for( int i = 0; i < n; i++ )
15. cin>>s[i];
17. int l = (n >> 1) - 1;
18. int r = (n & 1) ? l + 2 : l + 1;
19. *// cout<<l<<r<<endl;*
20. string res = "YES";
21. for( int i = 0; i < n; i++ )
22. {
23. for( int j = l, k = r; j >= 0; j--, k++ )
24. {
25. if( s[i][j] != s[i][k] || s[j][i] != s[k][i] )
26. {
27. res = "NO";
28. break;
29. }
30. }
31. if( res == "NO" ) break;
32. }
33. cout<<res<<endl;
34. }
35. return 0;
36. }

C#:

1. using System;
2. using System.Numerics;
3. class MyClass {
4. static void Main(string[] args) {
5. */\**
6. *\* Read input from stdin and provide input before running*
7. *var line1 = System.Console.ReadLine().Trim();*
8. *var N = Int32.Parse(line1);*
9. *for (var i = 0; i < N; i++) {*
10. *System.Console.WriteLine("hello world");*
11. *}*
13. *\*/*
15. var line1 = System.Console.ReadLine().Trim();
16. var N = Int32.Parse(line1);
17. var t=0; *//no of test cases*
18. short size;
19. string[] nos;
21. while(t<N){
22. line1 = System.Console.ReadLine().Trim();
23. size = Int16.Parse(line1); *//image dims*
24. short[,] img = new short[size, size];
25. for(var i =0; i<size; i++){
27. *//input img*
28. line1 = System.Console.ReadLine().Trim();
30. for(var j=0; j<size; j++){
31. img[i,j] = Int16.Parse(line1[j].ToString());
32. }
33. }
35. *//check for symmetry*
36. bool flag = true;
37. *//along y axis*
38. for(var i=0; i<size; i++){
39. for(var j=0; j<size/2; j++){
40. if(img[i,j]!=img[i,(size-1-j)]){
41. flag= false;
42. break;
43. }
44. }
45. }
46. if(flag)
47. *//check along x axis*
48. for(var i=0; i<size; i++){
49. for(var j=0; j<size/2; j++){
50. if(img[j,i]!=img[(size-1-j),i]){
51. flag= false;
52. break;
53. }
54. }
55. }
57. if(flag){
58. System.Console.WriteLine("YES");
59. }
60. else
61. System.Console.WriteLine("NO");
62. t++;
63. }
65. }
66. }

Java:

1. */\* IMPORTANT: Multiple classes and nested static classes are supported \*/*

4. *// uncomment this if you want to read input.*
5. import java.io.BufferedReader;
6. import java.io.InputStreamReader;

9. class TestClass {
10. public static void main(String args[] ) throws Exception {
12. BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
13. String line = br.readLine();
14. int t = Integer.parseInt(line);
15. for(int i=0;i<t;i++){
16. line = br.readLine();
17. int n = Integer.parseInt(line);
18. System.out.println((check(n,br)?"YES":"NO"));
19. }
20. }
22. static boolean check(int n,BufferedReader br) throws Exception{
23. String s[] = new String[n];
24. boolean flag=true;
25. for(int i=0;i<n;i++){
26. s[i] = br.readLine();
27. }
28. for(int i=0;i<n;i++){
29. if(!checkColumn(n,s[i])){
30. flag = false;
31. break;
32. }
33. }
34. if(flag && checkRows(s)){
35. return true;
36. }
37. return false;
38. }
40. static boolean checkColumn(int n, String s){
41. int l=0;
42. int r=0;
44. if(n%2 == 0){
45. l=(n/2)-1;
46. r=(n/2);
47. }else{
48. l=(n/2);
49. r=(n/2);
50. }
51. for(int i=l;i>=0 && r<n;i--){
52. if((s.charAt(l) != s.charAt(r))){
53. return false;
54. }
55. l--;
56. r++;
57. }
58. return true;
59. }
61. static boolean checkRows(String s[]){
62. int length= s.length;
63. int l=0;
64. int r=0;
66. if(length%2 == 0){
67. l=(length/2)-1;
68. r=(length/2);
69. }else{
70. l=(length/2);
71. r=(length/2);
72. }
73. for(int i=l;i>=0 && r<length;i--){
74. if(!s[l].equals(s[r])){
75. return false;
76. }
77. l--;
78. r++;
79. }
80. return true;
81. }
83. }

Java 8:

1. */\* IMPORTANT: Multiple classes and nested static classes are supported \*/*
3. */\**
4. *\* uncomment this if you want to read input.*
5. *//imports for BufferedReader*
6. *import java.io.BufferedReader;*
7. *import java.io.InputStreamReader;*
8. *\*/*
9. *//import for Scanner and other utility classes*
10. import java.util.\*;
11. import java.io.IOException;
12. import java.io.InputStream;
14. class TestClass3 {
15. public static void main(String args[] ) throws Exception {
16. */\**
17. *\* Read input from stdin and provide input before running*
18. *\* Use either of these methods for input*
20. *//BufferedReader*
21. *BufferedReader br = new BufferedReader(new InputStreamReader(System.in));*
22. *String line = br.readLine();*
23. *int N = Integer.parseInt(line);*
24. *\*/*
26. *//Scanner*
27. Scan s = new Scan();
28. int T = s.scanInt();
30. for (int t = 0; t < T; t++) {
31. int N = s.scanInt();
32. boolean[][] m = new boolean[N][N];
33. for (int i = 0; i < N; i++) {
34. for (int j = 0; j < N; j++) {
35. m[i][j] = s.scanBoolean();
36. }
37. }
39. boolean match = true;
40. for (int i = 0; i <= N/2; i++) {
41. for (int j = 0; j <= N/2; j++) {
42. if (m[i][j] != m[i][N-j-1] || m[i][j] != m[N-i-1][j] || m[i][j] != m[N-i-1][N-j-1]) {
43. match = false;
44. break;
45. }
46. }
47. }
49. if (match) {
50. System.out.println("YES");
51. }
52. else {
53. System.out.println("NO");
54. }
55. }
56. }
57. }
59. class Scan {
60. private byte[] buf=new byte[1024];
61. private int index;
62. private InputStream in;
63. private int total;
64. public Scan() {
65. in=System.in;
66. }
68. public int scan()throws IOException {
69. if(total<0) {
70. throw new InputMismatchException();
71. }
72. if(index>=total) {
73. index=0;
74. total=in.read(buf);
75. if(total<=0) {
76. return -1;
77. }
78. }
79. return buf[index++];
80. }
82. public int scanInt()throws IOException {
83. int integer=0;
84. int n=scan();
85. while(isWhiteSpace(n)) {
86. n=scan();
87. }
88. int neg=1;
89. if(n=='-') {
90. neg=-1;
91. n=scan();
92. }
93. while(!isWhiteSpace(n)) {
94. if(n>='0'&&n<='9') {
95. integer\*=10;
96. integer+=n-'0';
97. n=scan();
98. }
99. else throw new InputMismatchException();
100. }
101. return neg\*integer;
102. }
104. public long scanLong()throws IOException {
105. long integer=0;
106. int n=scan();
107. while(isWhiteSpace(n)) {
108. n=scan();
109. }
110. int neg=1;
111. if(n=='-') {
112. neg=-1;
113. n=scan();
114. }
115. while(!isWhiteSpace(n)) {
116. if(n>='0'&&n<='9') {
117. integer\*=10;
118. integer+=n-'0';
119. n=scan();
120. }
121. else throw new InputMismatchException();
122. }
123. return neg\*integer;
124. }
126. public double scanDouble()throws IOException {
127. double doub=0;
128. int n=scan();
129. while(isWhiteSpace(n))
130. n=scan();
131. int neg=1;
132. if(n=='-') {
133. neg=-1;
134. n=scan();
135. }
136. while(!isWhiteSpace(n)&&n!='.') {
137. if(n>='0'&&n<='9') {
138. doub\*=10;
139. doub+=n-'0';
140. n=scan();
141. }
142. else throw new InputMismatchException();
143. }
144. if(n=='.') {
145. n=scan();
146. double temp=1;
147. while(!isWhiteSpace(n)) {
148. if(n>='0'&&n<='9') {
149. temp/=10;
150. doub+=(n-'0')\*temp;
151. n=scan();
152. }
153. else throw new InputMismatchException();
154. }
155. }
156. return doub\*neg;
157. }
159. public String scanString()throws IOException {
160. StringBuilder sb=new StringBuilder();
161. int n=scan();
162. while(isWhiteSpace(n)) {
163. n=scan();
164. }
165. while(!isWhiteSpace(n)) {
166. sb.append((char)n);
167. n=scan();
168. }
169. return sb.toString();
170. }
172. public byte[] scanBytes(int N)throws IOException {
173. byte[] bytes = new byte[N];
174. int diff = total-index;
175. int i=index;
176. for (;i<total && i-index < N; i++) {
177. bytes[i-index] = buf[i];
178. }
179. index = i;
180. if (diff >= N) {
181. return bytes;
182. }
183. in.read(bytes, diff, N-diff);
184. return bytes;
185. }
187. public boolean scanBoolean() throws IOException {
188. int integer=0;
189. int n=scan();
190. while(isWhiteSpace(n)) {
191. n=scan();
192. }
193. return n == '1';
194. }
196. private boolean isWhiteSpace(int n) {
197. if(n==' '||n=='\n'||n=='\r'||n=='\t'||n==-1) {
198. return true;
199. }
200. return false;
201. }
202. }

JavaScript(Rhino):

1. *// Below is a sample code to process input from STDIN.*
2. *// Equivalent in effect to the Java declaration import java.io.\*;*
3. importPackage(java.io);
4. importPackage(java.lang);
5. importPackage(java.math);
6. importPackage(java.util);
8. var sc = new Scanner(System['in']);
10. var numCases = sc.nextLine();
11. for (var i = 0; i < numCases; i++) {
12. *// start a case*
13. var done = false;
14. var dimension = sc.nextLine();
15. var matrix = new Array(dimension);
16. *// check the row*
17. for (var r = 0; r < dimension; r++) {
18. var line = sc.nextLine() + "";
19. matrix[r] = new Array(dimension);
20. for (var c = 0; c < dimension; c++) {
21. matrix[r][c] = line[c];
22. }
23. }
24. for (var row = 0; row <= Math.floor(dimension/2) && !done; row++) {
25. for (var col = 0; col <= Math.floor(dimension/2) && !done; col++) {
26. if (matrix[row][col] != matrix[row][dimension - col - 1] ||
27. matrix[row][col] != matrix[dimension - row - 1][col]) {
28. print ("NO");
29. done = true;
30. }
31. }
32. }
33. if (!done) {
34. print ("YES");
35. }
36. *// now check the columns*
37. }
38. *// End of input processing code.*

JavaScript(Node.js):

1. function read(input, callback) {
3. var entry = input.split('\n');
4. var cases = entry.shift();
5. var matrix, size, index;
7. while(cases-- > 0) {
9. size = entry.shift();
10. matrix = entry.splice(0, size);
12. for(index = 0; index < size; index++) {
13. matrix[index] = matrix[index].split('');
14. }
16. callback(matrix, size);
18. }
20. }
22. function isSymmetric(matrix, size) {
24. var count = Math.floor(size / 2);
25. var x1, x2, y1, y2, i;
27. while(count-- > 0) {
29. x1 = y1 = count;
30. x2 = y2 = size - count - 1;
32. for(i = 0; i < size; i++) {
33. if(matrix[y1][i] !== matrix[y2][i] || matrix[i][x1] !== matrix[i][x2]) {
34. return false;
35. }
36. }
38. }
40. return true;
42. }
44. function resolver(value) {
45. return (value? 'YES': 'NO') + '\n';
46. }
48. function main(input) {
49. read(input, function(matrix, size) {
50. process.stdout.write(resolver(isSymmetric(matrix, size)));
51. });
52. }
54. process.stdin.resume();
55. process.stdin.setEncoding("utf-8");
56. var stdin\_input = "";
58. process.stdin.on("data", function (input) {
59. stdin\_input += input;
60. });
62. process.stdin.on("end", function () {
63. main(stdin\_input);
64. });

Pascal:

1. uses crt;
2. var
3. x,t,n,i,j:longint;
4. a:array[1..32] of string;
5. syn:boolean;
7. procedure optimize;
8. begin
9. readln(n);
10. for i:=1 to n do
11. readln(a[i]);
13. syn:=true;
14. for i:=1 to (n div 2)+1 do
15. for j:=1 to (n div 2)+1 do
16. if (a[i][j]<>a[n-i+1][j]) or (a[i][j]<>a[i][n-j+1]) or (a[i][j]<>a[n-i+1][n-j+1])
17. then syn:=false;
18. if syn=true then writeln('YES') else writeln('NO');
19. end;
21. procedure input;
22. begin
23. readln(t);
24. for x:=1 to t do
25. begin
26. optimize;
27. end;
28. end;
30. begin
31. input;
32. end.

Perl:

1. use strict;
2. my $testcase = <>;
3. for (my $i=1; $i<=$testcase; $i++) {
4. my $matrix = <>;
5. my $input;
6. my $flag=0;
7. my $temp=0;
8. my @input;
9. for (my $j=1; $j<=$matrix; $j++) {
10. $input = <>;
11. chomp $input;
12. push (@input,$input);
13. }
14. my $len=@input;
15. for (my $k=0; $k<$len; $k++) {
16. if ($input[$k] == $input[-($k+1)]) {
17. $temp++
18. }
19. }
20. foreach (@input) {
21. if ($\_ == reverse($\_)) {
22. $flag++;
23. }
24. }
25. if (($temp==$len) && ($flag==$len)) {
26. print "YES\n";
27. } else {print "NO\n";}
28. }

PHP:

1. <?php
3. *// Sample code to perform I/O:*
5. fscanf(STDIN, "%s\n", $total\_chance); *// Reading input from STDIN*
6. *// Writing output to STDOUT*
8. *// Warning: Printing unwanted or ill-formatted data to output will cause the test cases to fail*
10. for($k=0;$k<$total\_chance;$k++)
11. {
13. $metrix = array();
14. fscanf(STDIN, "%s\n", $no\_row);
16. $n= $no\_row-1;
18. for($s=0;$s<$no\_row;$s++){
20. fscanf(STDIN, "%s\n", $row);
22. for($l=0;$l<$no\_row;$l++){
24. $metrix[$s][$l]=$row[$l];
25. }
27. }

30. *//start*
31. $symetric = 'YES';
32. for($i=0; $i<=$n;$i++){

35. for($j=0;$j<=$n;$j++){
37. if($metrix[$i][$j]!=$metrix[$i][$n-$j]){
38. $symetric = 'NO';break 2;
39. }
41. if($metrix[$i][$j]!=$metrix[$n-$i][$j]){
42. $symetric = 'NO';break 2;
43. }
45. }

48. }
50. echo $symetric.PHP\_EOL;
52. *//end*
54. }

57. *// Write your code here*
59. ?>

Python:

1. '''
2. # Read input from stdin and provide input before running code
4. name = raw\_input()
5. print 'Hi, %s.' % name
6. '''
8. for \_ in range(input()):
9. def sym():
10. n = input()
11. g = [raw\_input() for \_ in range(n)]
12. for i in range(n):
13. for j in range(n):
14. if g[i][j] != g[-1-i][j] or g[i][j] != g[i][-1-j]:
15. return False
16. return True
17. print 'YES' if sym() else 'NO'

Python 3:

1. for testcase in range(int(input())):
2. n = int(input())
3. matrix = [input().strip() for \_ in range(n)]
4. *# check top bottom*
5. if n % 2 == 0:
6. top, bottom = matrix[:n*//2], matrix[n//2:]*
7. else:
8. top, bottom = matrix[:n*//2], matrix[(n//2)+1:]*
9. bottom.reverse()
10. topok = top == bottom
11. if not topok:
12. leftok=False
13. else:
14. *# Check left right*
15. leftok = True
16. for row in matrix:
17. if n%2 == 0:
18. left, right = row[:n*//2], row[n//2:]*
19. else:
20. left, right = row[:n*//2], row[(n//2)+1:]*
21. left, right = list(left), list(right)
22. right.reverse()
23. if left != right:
24. leftok = False
25. break
26. if topok and leftok:
27. print('YES')
28. else:
29. print('NO')

Ruby:

1. t = gets.to\_i
2. t.times do
3. n = gets.to\_i
4. matrix = []
5. output = "YES"
6. n.times do
7. line = gets.chomp
8. *#puts "#{line == line.reverse}, line = #{line}"*
9. output = 'NO' if !(line == line.reverse)
10. matrix << line.split(' ').map(&:to\_i)
11. end
12. if output == "NO"
13. puts output
14. next
15. else
16. *#puts "matrix"*
17. *#puts matrix*
18. (0..n-1).each do |i|
19. if !(matrix[i]==matrix[n-1-i])
20. output = "NO"
21. break
22. end
23. end
24. puts output
25. end
26. end