Project description:

Language used	Python
Version used	2.7.5
Dependencies	Python 2.7
Make File	Since I have used python, command make directly runs the script

Time frame of this project:

Actual vs Planned:

Task	**Planned (in hours)**	**Actual (in hours)**
Planning – Understanding the requirements	1	1:30
Mathematical Understanding of the problem	1	2:00 [time took in expressing 3rd constraint]
Design- study on how to implement specific parts of the project	00:30	00:20
Implementation	00:45	1:00
Testing the code	00:15	00:30

Observations:

1. Since half the class consists of smart and non-smart students the value of 'N' cannot be odd as this would make N*N odd.

2. Let N=2*k => total number of students is $4k^2$
Ratio of smart: non smart students is same across all the rows.
Let no. of smart students in a row be r_1. r_1:N- r_1 is the desired ratio.
We know that the total number of smart students=$2k^2$. r_1 students in one row

=> r_1 * 2k students in 2k (N) rows.
r_1*2k=$2k^2$ => r_1=k

=> **<u>ratio of smart: non smart students in each row= 1:1</u>**

3. Constraint that all the columns must have different ratio implies that each column must have different number of smart students.

Note: Two columns having different number of smart students cannot have same ratio. Proof by contradiction:

Assume two columns with corresponding number of smart students as C_1 and C_2, have same ratios:

=> C_1: N-C_1 = C_2:N-C_2
=> (N-C_2)*C_1 =(N-C_1)*C_2
=> C_1=C_2 which contradicts our assumption
Hence two columns with different number of smart students have different ratios. Therefore the constraint boils down to filling each column with different number of smart students.

Let C_i be the number of smart students in column i. We know that total number of smart students is 2K^2. So sum of smart students across all the columns must be 2K^2

=> $\sum C_i$ = 2K^2 . for i in [1,2k] . 0<=C_i <=2k

Here sum of unique integers from 0 to 2k is 2k^2. Only one integer misses out from this range.

Let the number be 'e'. Hence sum of integers from 0 to 2k without 'e' is 2k^2.

=>2K*(2K+1)/2 - e = 2k^2
=>2k^2 + k -e =2k^2

e=k, the number that is going to miss out is k => column cannot have 1:1 ratio.
So each column must draw their cardinality of smart students without replacement from [0,1…, 2*k] –{k}

Algorithm:

Algorithm ConstructMatrix(N):
Input: Order of the square matrix
Output : N*N matrix containing arrangement of smart and non-smart students
Row_track :=list of size N each element being N/2 [to ensure that each row has N/2 smart students]
Column_track:=list of size N containing [0,1…, 2*k] –{k} [maintains the number of smart students in each column]
a= N*N matrix with each element as 0
// The matrix is constructed column wise

```
for j in range(0,N):
    unmark all rows;
    while column_track[j]>0:
            if upper diagonal position is available
                then curr_pos=upper_diagonal
            else if lower diagonal position is available
                then curr_pos=lower_diagonal
            else
                Greedily choose the available row with maximum vacancy
            a[curr_pos][j]=1;
            mark curr_pos row; // make it unavailable
            Column_track[i]--;
            row_track[curr_pos]--;
Output matrix 'a' such that  0 is encoded as 'N' and 1 as 'S'
```

Project execution:

```
$ make
```

This directly executes the python script and prompts for an input for N. Supply the appropriate input.

Sample input and output:

```
Please enter the value of N: 11

Odd numbers not applicable, please enter an even number: 1

Odd numbers not applicable, please enter an even number: 6

N S N S N S
N S N S S N
N S S S N N
N S N S N S
N S N N S S
N S N S N S
```