Problem

In this lab, you are going to implement functions related to some basic matrix operations of which prototypes given below. You are not going to define a "main" function. Instead, your functions will be evaluated with our "main" function.

At the top of your code include the following header: "lab.h"

```
#include "lab.h"
```

Do not define a "main" function in your code! You should define a function named "execute" which will be called when you execute your program. Note that this function will not be called when evaluating your program. It is provided only to enable you to test your functions manually.

```
int execute()
{
	/* your test code */
	return 0;
}
```

"lab.h" contains a 10x10 global array named "matrix". You are supposed to use this array in your functions. Do not define a new global array!

```
#define N 10
int matrix[N][N];
```

To ease things up for you, here is a sample code that defines a function to print the values of the array:

```
#include <stdio.h>
#include "lab.h"

int execute() {
	/* insert code to test your functions */
	return 0;
}

void printMatrix() {
	int i, j;
	for(i=0; i<N; i++){
		for(j=0; j<N; j++)
			printf("%d ", matrix[i][j]);
		printf("\n");
	}
}
```

You may start building your code using the sample code above.

You are supposed to define and implement the following functions, all of which operate on the global matrix defined in "lab.h". Make sure you define all the functions; otherwise your code will not compile. Even if you do not implement any one of the functions, provide an empty definition. For instance;

```
void printMatrix() {
        /* not implemented, but defined */
}
```

Your program will be evaluated function-by-function so you may get partial credits even if you do not implement all the functions correctly.

Debugging

You can use the "printf" function to DEBUG your code. After each printf call, you need to use "fflush" function to see the output on the screen:

```
printf("i am here\n");
fflush(stdout);
```

----

void readRowByRow(void);

This function reads an NxN matrix row-by-row. When the function is called, it will wait for the user to enter the matrix in row-by-row manner. From the terminal, content of each row will be provided as a separate line. The first number in each line corresponds to the row number which the line belongs to.

Example: For N=4,

```
2 1 1 2 4
0 2 4 2 3
3 4 5 6 3
1 2 1 0 1
```

will construct 4x4 matrix below:

```
2 4 2 3
2 1 0 1
1 1 2 4
4 5 6 3
```

The first line of the input starts with 2, which means that the remaining numbers, (1 1 2 4) belong to the 2nd row of the matrix. Inspect the output of the sample run above for more information.

Note: This function does not print anything to the screen; it just changes the contents of the global matrix.

----

void printColumnByColumn(void);

This function prints an NxN matrix column by column. It will print the matrix so that each column will be printed on a line.

Example: For the 4x4 matrix that we constructed before, output will be:

2 2 1 4
4 1 1 5
2 0 2 6
3 1 4 3

----
void printSnakelike(void);

This function prints an NxN matrix in a snakelike form. It will print the first row in order, print second row in reverse order, print the third row in order etc ...

Example: For the 4x4 matrix,
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
output of the function will be:
1 2 3 4
8 7 6 5
9 10 11 12
16 15 14 13

----
int rowSum(int rowNumber);

This function returns the summation of the numbers in a given row.

Example: For the 4x4 matrix; if the function is called with 1 as the argument,
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
the function will return the sum of the 1st row (5 6 7 8), which is 5+6+7+8 = 26.

Note: This function does not print anything to the screen; it just returns the summation as an integer.

----
void printEachSum(void);

This function prints the summation of the numbers line-by-line for each row.

Example: For the 4x4 matrix,
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
the function will print:
10
26

42
58

Hint: Use "rowSum" function defined earlier.

----
void transpose(void);

This function will take the transpose of the matrix. The transpose of a matrix is calculated by swapping matrix[i] and matrix[j] for each i, 0 <= i < N and j, 0 <= j < N.

Example: For the 4x4 matrix,
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
the function will change the contents of the matrix so that matrix will contain the following numbers:
1 5 9 13
2 6 10 14
3 7 11 15
4 8 12 16

Note: This function does not print anything to the screen; it just changes the contents of the global matrix.
