LAB 11: Function with array and pointer

จงเขียนฟังก์ชันต่างๆ ต่อไปนี้ให้สมบูรณ์ โดย...

ฟังก์ชัน GetMatrix ใช้สำหรับรับค่า Matrix พร้อมจำนวนแถวและคอลัมน์

ฟังก์ชัน MatrixTranspose ใช้กลับค่าแมทริกซ์จากแถวเป็นคอลัมน์จากคอลัมน์เป็นแถว

ฟังก์ชัน MatrixMultiply ใช้คูณแมทริกซ์

ฟังก์ชัน PrintMatrix พิมพ์ค่าในแมทริกซ์

เพื่อให้สามารถเรียกใช้ด้วยฟังก์ชัน main สำหรับดำเนินการกับแมทริกซ์ดังนี้

1. ประกาศตัวแปร M1, M2, M3, M4 เป็นตัวแปรแอเรย์ 2 มิติของเลขจำนวนเต็ม
2. ประกาศตัวแปร r1, c1 สำหรับเก็บจำนวนแถวและคอลัมน์ของ M1
3. ประกาศตัวแปร r2, c2 สำหรับเก็บจำนวนแถวและคอลัมน์ของ M2
4. ประกาศตัวแปร r3, c3 สำหรับเก็บจำนวนแถวและคอลัมน์ของ M3
5. ประกาศตัวแปร r4, c4 สำหรับเก็บจำนวนแถวและคอลัมน์ของ M4
6. เรียกฟังก์ชัน GetMatrix เพื่อรับค่าของ M1, r1, c1
7. เรียกฟังก์ชัน GetMatrix เพื่อรับค่าของ M2, r2, c2
8. เรียกฟังก์ชัน TransposeMatrix เพื่อหาค่า M3 จากการ Transpose ค่าของ M1
9. เรียกฟังก์ชัน MatrixMultiply เพื่อหาค่า M4 จากการ คูณ M1 กับ M2
10. เรียกฟังก์ชัน PrintMatrix เพื่อพิมพ์ค่าของทุกแมทริกซ์

ตัวอย่างผลลัพธ์หน้าจอ

```
Enter Matrix 1:
Enter number rows and columns: 3 2
Enter [0, 0] value: 2
Enter [0, 1] value: 3
Enter [1, 0] value: 3
Enter [1, 1] value: 1
Enter [2, 0] value: 2
Enter [2, 1] value: 4

Enter Matrix 2:
Enter number rows and columns: 2 4
Enter [0, 0] value: 5
Enter [0, 1] value: 2
Enter [0, 2] value: 1
Enter [0, 3] value: 3
Enter [1, 0] value: 2
Enter [1, 1] value: 4
Enter [1, 2] value: 1
Enter [1, 3] value: 2

Matrix 1:
 2  3
 3  1
 2  4

Matrix 2:
 5  2  1  3
 2  4  1  2

Transpose of Matrix 1 is:
 2  3  2
 3  1  4

Matrix1 x Matrix2 is:
16 16  5 12
17 10  4 11
18 20  6 14
```

LAB 11: Function with array and pointer

เติมโปรแกรมให้สมบูรณ์

```
#include<stdio.h>
#include<math.h>

void GetMatrix(int M[][5], int *rows, int *columns)
{
	int i,j;
	printf("Enter number rows and columns: ");
	scanf("%d %d", _______________, c_______________);
	for(i=0;i< __________;i++)
		for(j= 0; j< _____________;j++)
		   {
			printf("Enter [%d, %d] value: ", i,j);
			scanf("%d", _________________);
		    }
}

void MatrixTranspose(int M[][5], int rows, int columns, int Result[][5], int *ResultRows,
				int *ResultColumns)
{
	int i,j;
	_________________ = columns;
	_________________ = rows;
	for(i=0;i<_____________;i++)
		for(j=0;j<_______________;j++)
			Result[i][j] = ____________________________;
}

int MatrixMultiply(int M1[][5], int rows1, int columns1, int M2[][5], int rows2, int columns2,
		  int Result[][5], int *ResultRows, int *ResultColumns)
{
	int i,j,k;
	if(columns1!=rows2)
		return 0;		/* คืนค่าเป็น 0 ถ้าคูณไม่ได้ */
	else
	{
		*ResultRows = ____________________________;
		*ResultColumns = ____________________________;
		for(i=0;i<_____________________;i++)
			for(j=0;j<_____________________;j++)
			{
				Result[i][j] = 0;
				for(k=0;k<columns1; k++)
					Result[i][j] += ____________________________
			}
		return 1;		/* คืนค่าเป็น 1 ถ้าคูณได้ */
	}
}
```

LAB 11: Function with array and pointer

```
void PrintMatrix(int M[][5], int rows, int columns)
{
        int i,j;
        for(i=0;i<rows;i++)
        {
                for(j=0;j<columns;j++)
                        _____________________________________________;
                printf("\n");
        }
}


void main(void)
{
    int M1[5][5], M2[5][5], M3[5][5], M4[5][5];
    int r1, c1, r2, c2, r3, c3, r4, c4;

    printf("Enter Matrix 1: \n");
    GetMatrix(M1, &r1, &c1);
    printf("Enter Matrix 2: \n");
    GetMatrix(M2, &r2, &c2);

    printf("Matrix 1:\n");
    PrintMatrix(M1, r1, c1);
    printf("Matrix 2:\n");
    PrintMatrix(M2, r2, c2);

    MatrixTranspose(M1, r1, c1, M3, &r3, &c3);
    printf("Transpose of Matrix 1 is: \n");
    PrintMatrix(M3, r3, c3);

    if( MatrixMultiply(M1, r1, c1, M2, r2, c2, M4, &r4, &c4) == 0)
        printf("Matrix1 and Matrix2 cannot multiply\n");
    else
   {
        printf("Matrix1 x Matrix2 is: \n");
        PrintMatrix(M4, r4, c4);
    }
}  /* End of main */
```