MATH 261, Spring 2001 **Due Date:**_____ **Name(s):** _____

Honors Project 5: Painting Arrays with Nested Do-Loops

Objective

In this project we discuss the "painting" of 2-dimensional arrays using "nested do-loops" in anticipation of studying multiple integration over regions of arbitrary shape, and convolution and digital image processing.

Narrative

To do `<yada>` 3 times in Maple, we may use the *do-loop* command:

```
for n from 1 to 3 do <yada> od;
```

If `<yada>` is itself a do-loop, the resulting composite command is known as a *nested do-loop*. Nested do-loops are an important programming construct. Since the limits on the inner do-loop of a nested do-loop can depend on the counter of the outer do-loop, we can "paint" a rectangular array — or *matrix* — of 0's with 1's in some interesting ways. In this project we investigate this process in anticipation of studying multiple integration over regions of arbitrary shape.

Task

1. a) Type the command lines below into Maple in the order in which they are listed; they define a 6×12 square array `M` of 0's, fill — or "paint" — part of this array with 1's, and then add up all the entries of the array (so that `Tally` eventually represents the number of 1's in `M`) ... first over the *entire* array `M` and then over only the nonzero entries of `M`.

```
> # Project 05:  Painting Arrays
> restart;
> with(linalg):
> # Declare M as a 6x12 array
> M := matrix(6,12,(Row,Col) -> 0);
> # Task 1a
> # Paint M
> for Row from 1 to 6 do
   for Col from 1 to Row do M[Row,Col] := 1 od; od;
> evalm(M);
> # Check 1
> Tally := 0;
> for Row from 1 to 6 do
   for Col from 1 to 12 do Tally := Tally + M[Row,Col] od; od;
> Tally;
> # Check 2
> Tally := 0;
> for Row from 1 to 6 do
   for Col from 1 to Row do Tally := Tally + M[Row,Col] od; od;
> Tally;
```

b) Repeat Task 1a, replacing the code, "`for Col from 1 to Row do M[Row,Col] := 1 od; od;`" with, "`for Col from Row to 13-Row do M[Row,Col] := 1 od; od;`".

2. For each of the following arrays, use a pair of nested do-loops — as in Task 1 — to create the following arrays from the zero array, and check your work as you did in Task 1:

0	1	1	1	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	1	1	0	0
0	0	0	1	1	1	1	1	1	1	0	0
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	0	0	1	1	1	1	1	0	0
0	0	0	0	0	0	1	1	1	1	0	0

0	1	1	1	1	1	1	1	1	1	1	1	0
0	0	1	1	1	1	1	1	1	1	1	0	0
0	0	0	1	1	1	1	1	1	1	0	0	0
0	0	0	0	1	1	1	1	1	0	0	0	0
0	0	0	0	0	1	1	1	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0

0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	1	1	1	0	0	0	0	0
0	0	0	0	1	1	1	1	1	0	0	0	0
0	0	0	1	1	1	1	1	1	1	0	0	0
0	0	1	1	1	1	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	0
0	0	1	1	1	1	1	1	1	1	1	0	0
0	0	0	1	1	1	1	1	1	1	0	0	0
0	0	0	0	1	1	1	1	1	0	0	0	0
0	0	0	0	0	1	1	1	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0