Melbourne University
Mathematics and Statistics Research Competition 2020

Presbyterian Ladies' College
Question 2 - Colouring Squares

Team Members: Liah Wu, Yunaa Tae

July 26, 2020

CONTENTS

1. Question 2 - Colouring Squares Report 1

2. Program and Results 17

    2.1 Program Detail 17

    2.2 Program Source Code 17

    2.3 Test Cases and Results 18

Bibliography 19

Appendix 21

1. Question 2 - Colouring Squares

The Problem

For an $m \times n$ grid, and $k \geq 2$ colours, how many ways are there to colour a grid so that no neighbouring* squares are the same colour?

**Neighbouring squares are squares immediately to the left and right, or directly above and below, the original square.*

Solution

<u>Different cases</u>

We will solve the problem algebraically, moving in one direction, from the top left square to the bottom right square. We will move down the left most column, then go back up into the second column from the left, and start at the top square, but never move to the left or upwards. Also, the number of possible colours that a square could be will be written in the square itself, and the number of possibilities is dependent on the squares already filled with some number of colours.

To start off, look at a $1 \times n$ grid, as shown below.

k	k - 1	k - 1	k - 1	k - 1

Say there are k colours. Then going from left to right, the first square (left-most) would have k possible colours. The second square, which cannot be filled with the same colour as the first square, can be filled with (k-1) colours. Same goes to the third square, which could be filled with any of the k colours except for the colour to the left (which is the second square) which means the total possible colours is also (k-1). Following this idea, for a $1 \times n$ grid, there are (n-1) values of (k-1), and as the first square has k possibilities, multiplied by the k possibilities for the first square, giving, $k(k - 1)^{n - 1}$ ways to colour a $1 \times n$ grid.

However, when m>1, there can be 2 general cases for the number of solutions possible. They are shown below.

Note: When saying "diagonally adjacent" we mean a square and the square that is diagonally opposed to it (sharing only 1 vertex, but no edges), as shown in the green and blue coloured squares in the grid below. Also, this case only exists when k≥*3, as the diagonally adjacent colours must be both different to each other, and different to the square to the left/top respectively, as shown in yellow in the diagram below.*

Case 1

No squares on the grid which are diagonally adjacent are coloured the same.

We will use a 4 × 3 grid to show this. As proved above, the number of combinations for the first column is k for the left-most square, and then (k - 1)for the following squares, as shown below.

k			
k - 1			
k - 1			

Now, going down to the second column, the top square on the second column couldn't be the same colour as the top-left block, or the diagonally adjacent square to the bottom-left. This gives (k - 2) possibilities for this square. The second square from the left on the second row could not be the colour of the square above, next to, or diagonally adjacent to (two squares since from the squares of the colours we already determined, both of which are in the first row, the top-left and bottom-left), so this square could be coloured in (k-4) ways. This goes all the way until we reach the third and last row. When we get to the last row, the only squares that affect the one we're on about are the squares to the top, left and the diagonally adjacent square to the top-left, so this then gives

(k-3) possible colourings. Repeating this process with the next row gives the same pattern of colouring, as shown below.

k	k - 2	k - 2	k - 2
k - 1	k - 4	k - 4	k - 4
k - 1	k -3	k - 3	k - 3

But after some thinking, you will realise that there are a lot of mistakes in multiplying this all together. Look at the grid below, and focus on the square x. Before, we assumed that there were (k - 4) possibilities, but we know that while $b \neq x$, $d \neq x$, we do not know that $b \neq d$, so while x might be (k - 4), it could also be (k - 3), and if x was (k- 3), the squares near it would have another value, since it means that b and d would have to be equal.

b	c	d	
a	x		

From this, it is evident that there is no such obvious formula for Case 1 grids, and even if there was a formula, it would be quite unpleasant.

Case 2

There are squares on the grid which are diagonally adjacent and are coloured the same.

The *2×2* grid shown below is a typical case of a pair of diagonally adjacent squares coloured the same.

	a
a	

In a general case, this is what it would look like (in terms of possible colourings):

k	1
k - 1	k - 1

This is as the first column, has one k and (k-1) as proven above. The top square in the second column would have 1 possibility, because it would have to be dependent upon the diagonally adjacent square that we have already determined has (k-1) possible colourings. No matter what the diagonally adjacent square is coloured, it would have to be the same colour. However, the bottom right square also has (k-1) possibilities, as the square to the top and left of it is the same colour, so that square can only not be of one colour, and hence (k-1) possibilities.

However, the formula is not straightforward either, because in the grid, the number of diagonally-adjacent squares being the same colour and and where they would be placed would vary, which would mean the formula would be an unpleasant one, if one was even possible. Therefore, a general solution must be formed.

The General Solution

To reach a general solution, we will look at the cases where $m = 1$, 2, 3 to get a sense of what could be done to form a general method, or hopefully a formula. Previously, we have already solved the case for an $m \times n$ graph if $m = 1$, so we will start at $m = 2$ and then move on to solve the case for if $m = 3$.

2×*n* grid

k_{a}	
k_{b}	

As explained above, the number of possibilities for the first column would be k(k-1). In the next column, the top square must be of a different colour as the square to the top-left, so in this case it must not be colour k_{a}. Now, let's split the colour choice of the top-right square into two cases - if it is k_{b} (the same colour as the square diagonally-opposite to the bottom-left) or if it is a different colour to k_{a} and k_{b}.

Case 1

If the top square of the second column was coloured with k_b as shown below.

k_{a}	k_{b}
k_{b}	

If the colour was k_{b}, then the square below could be $k_{\mathrm{a,c,d...}}$ etc, which is just any colour that is not k_{b}. This is shown below.

k_{a}	k_{b}
k_{b}	$k_{\mathrm{a,c,d...}}$

For the top-left square, you can choose any colour, hence there are k possibilities. Since in this case the squares on the top-right and bottom left are of the same colour, and are both adjacent to the top-left square, there are (k-1) possible colourings in total for both of those squares. For the bottom-right square, it is adjacent to two squares, however because of the case, they are of the same colour. Hence, the bottom right square also has (k-1) possibilities of colouring it. Hence, this gives the number of possibilities per square as follows.

k	1
(k-1)	(k-1)

Therefore, if the top square of the second column was coloured with k_{b}, there would be (k-1) possible colourings for the second column, and in a 2×2 grid case, there would be k(k-1)^2 possible colourings of the entire grid.

Case 2

If the top block of the second column was coloured with k_x (where $x \neq a, b$), as shown below.

k_{a}	$k_{\mathrm{x}\ (x \neq a, b)}$
k_{b}	

Say the top square of the second column was any other colour apart from k_{a} (because it is adjacent to a square using that colour) and k_{b} (not the same as case 1), this means that the block to the bottom of that square (the bottom-right square) could be any colour that wasn't the same colour as the one to its top or k_{b} (the square to the left), as shown below.

k_{a}	$k_{\mathrm{x}\ (x \neq a, b)}$
k_{b}	$(k_{\mathrm{a,c,d,...}})$

For the top-left square, you can choose any colour, hence there are k possibilities. Since in this case the squares on the top-right and bottom left are of different colours, the bottom square in the first column has then (k-1) possibilities, and because it's not of the same colour as the top-right square, the top-right square has (k-2) possibilities. The bottom-right square is adjacent to two squares, both of which in this case are of different colours, and hence also have (k-2) possibilities. This is shown below.

k	(k-2)
(k-1)	(k-2)

In case 2, the first column has k(k-1) possibilities, and the second column has (k-2)(k-2) possible colourings. Hence, there are (k-2)^2 possible colourings for the second column, if the top square wasn't $k_{\mathrm{b,a}}$, and we have the formula for a 2×2 grid, which is $k(k - 1)(k- 2)^2$.

Now, we are going to piece together the formula that covers the total cases for a 2×n grid. By looking at the 2×2 grid case, we discover the second column could be coloured in $((k - 1) + (k\text{-}2)^2) = k^2 - 3k + 3$ ways. By repeating the process above for a 2×3 grid, we discover that still the number of possibilities for the 3rd column would be (k^2 - 3k + 3). This is because the colouring of the next column would be only dependent on how the previous column was coloured, not on how any other columns could be coloured, so we could do this (n-1) times for a 2×n grid, and find the formula for total number of possible colourings of a 2×n grid as $k(k - 1)(k^2 - 3k + 3)^{n - 1}$.

$3\times n$ grid

Unlike the $2\times n$ grid, where there must be 2 different colours, a column could contain 2 or 3 different colours. Let a, b, c represent different colours. Then the first column could be in two forms: a-b-a, or a-b-c (going down the column). Call "a-b-c" "*Type* 1", t_1 for short, and "a-b-a" as "*Type* 2", t_2 for short. On the grid, if the previous column, or the column to its left is of t_1, then the next column could be of either form, so either of t_1 or t_2. Similarly, if the previous column was of form t_2, then the next column can also be of either form t_1 or t_2.

So first we must know how many possible combinations there are in the instance that a t_1 follows a t_1 or t_2 column, and also the number of possibilities for any t_2 that follows a t_1 or t_2 coloured column. Let m_{n} be the number of possible colourings of a *3*$\times n$ grid ending with a column of t_1 (ie. the right-most column in the grid is of type t_1), and let x_{n} be the number of possible colourings of a $3\times n$ grid that ends in a column of t_2 (ie the right-most column in the grid is of type t_2).

This means that $m_{\text{n+1}}$ would equal the total number of ways that go from a t_1 in the previous column to t_1 in the last column multiplied by the total number of possibilities m_{n}, plus the number of ways a t_2 is followed by a t_1, multiplied by the number of possibilities of x_{n}. Following a similar fashion, $x_{\text{n+ 1}}$ would be the number of ways a t_2 could follow a t_2 multiplied by x_{n}, plus ways to get from t_1 to t_2 multiplied by the number of possibilities m_n. As an equation, this would look like as shown below.

$m_{\text{n+1}} = (\text{ways t}_1 \rightarrow t_1)m_{\text{n}} + (\text{ways } t_2\rightarrow t_1)x_{\text{n}}$
$x_{\text{n+1}} = (\text{ways } t_2\rightarrow \text{t}_2)x_{\text{n}} + (\text{ways } t_1\rightarrow t_2)m_{\text{n}}$

To calculate, all of the above can be simplified into matrices. But first, we need to know the number of possible colourings for a t_1 that follows either a t_1 or t_2, and also the number of possibilities for any t_2 that follows a t_1 or t_2.

Note: When using '→', we mean the number of ways the column type to the left of the arrow could be added after the column type to the right of the arrow. ie. $t_1 \rightarrow t_1$ *means the number of ways a* t_1 *column could appear to the right of another* t_1 *column, to add another column to the already existing grid.*

$\boldsymbol{t_1 \rightarrow t_1}$

$$\begin{array}{|c|}\hline a \\ \hline b \\ \hline c \\ \hline \end{array} \rightarrow \begin{array}{|c|}\hline a \\ \hline b \\ \hline c \\ \hline \end{array}$$

There are 4 cases for when a t_1 follows a t_1 column.

<u>Case 1</u>
all 3 colours in the next column are different to the previous column (ie. not a, b or c)

That means the top block of the next column only cannot be a, b or c, and hence has k - 3 possible colours to choose from. The next one down also cannot be any of the 3 colours a, b or c, but also cannot be the same colour as the square above, so k - 4. The last block must not be the same colour as a, b, c and also not the same as the above 2 squares (as identified by this case), and hence have k - 5 possible colourings. This is shown below.

a	k - 3
b	k - 4
c	k - 5

Therefore, there are $(k - 3)(k - 4)(k - 5) = k^3 - 12k^2 + 47k - 60$ total combinations that satisfy the rules of case 1.

Case 2

2 colours in the next column are different

No matter where the same colour is placed in the next column, the remaining two squares can't be any of the three colours a, b or c, so there are k - 3 possibilities, and hence the last square has k - 4 possibilities.

Then, no matter which colour remained the same, it could be in 2 other positions, and the remaining 2 empty blocks could then be filled with (k-3)(k-4) = k^2 - 7k + 12 possible combinations of 2 other colours.

As for the repeated colour, there are 3 possible colours to choose from to repeat (a, b or c), and for each colour there are 2 possible places where it can be placed, as it cannot be placed next to one of the same colour in the previous column. Hence, there are $3 \times s \times (k-3)(k-4) = 6 \times (k^2 - 7k + 12) = 6k^2$ - 42k + 72 possibilities.

Case 3

only one colour in the next column is different

There would be three ways of choosing which the one different colour would be, and also there are three ways of choosing where the two of the original colours would be placed. The remaining block, as it must be a different colour to the original 3, has k - 3 possible ways of doing it. Now, multiplying the number of choices the 2 remaining colours could be and where they are placed, with the number of possibilities the remaining colour could be, there are a total of $9 \times (k-3) = 9k$ - 27 ways of colouring using this case.

Case 4
all colours in the next column are the same as the ones in the previous

When all three colours stay the same as the previous row (ie. using colours a, b and c ONLY), then there are only 2 ways of arranging the 3 colours that satisfy the rules. Those are b-c-a and c-a-b (from top to bottom). Hence, there are 2 solutions to this case.

Summing up the number of possibilities from a t_1 to another t_1 type colouring using the 4 cases above, we get k^3 - $12k^2$ + $47k$ - 60 + $6k^2$ - $42k$ + 72 + $9k$ - 27 + 2 = k^3 - $6k^2$ + $14k$ - 13 possibilities in total if we go from t_1 to t_1.

$\boldsymbol{t_1 \rightarrow t_2}$

$$\begin{array}{|c|}\hline a\\\hline b\\\hline c\\\hline\end{array} \rightarrow \begin{array}{|c|}\hline a\\\hline b\\\hline a\\\hline\end{array}$$

There are 3 cases possible when going from t_1 to t_2.

Case 1
all colours in the next column are different.

As there are only two colours used (a and b), then the two remaining squares have k - 3 and k - 4 different possibilities, since 3 colours are used in the previous t_1 column. This gives (k - 3)(k - 4) = k^2 - $7k$ + 12 possibilities.

Case 2
one colour in the next column is different to the ones in the previous column

This case can be split into 2 subcases. If the colour that is the same as the previous column was a or c, then a or c could only be placed in the middle, and the remaining colour could be either of the k - 3 that haven't been used. Since this occurs to both a and c, there are 2(k - 3) = 2k - 6 possibilities. If b was the colour remaining, then b could only be at the top and bottom, and the middle block could be filled with k-3 colours. Summing these up gives 2k - 6 + k - 3 = 3k - 9 possibilities.

Case 3
the colours in the next column are all from the previous column

As no colour can be next to the same colour, b must be placed on the top and bottom block, and since a or c can be placed in the middle block, there are 2 total possibilities.

Therefore, summing the 3 cases up, we get k^2 - 7k + 12 + 3k - 9 + 2 = k^2 - 4k + 5 possibilities in total.

$t_2 \rightarrow t_1$

$$\begin{array}{|c|}\hline a\\\hline b\\\hline a\\\hline\end{array} \rightarrow \begin{array}{|c|}\hline a\\\hline b\\\hline c\\\hline\end{array}$$

Case 1
all colours in the next column are different

Since there are only 2 colours in type t_2, columns, if the next column was to have all different colours, then the top square would have (k-2) possible colours, the next one would have (k-3) possible colours, while the last one would have (k-4) colours. Therefore, the total number of possible colourings would be (k - 2)(k - 3)(k - 4) = k^3 - 9k^2 + 26k - 24.

Case 2

two colours in the next column are different from the ones that appeared in the previous column

Say 'a' was the colour that remained the same. Then 'a' could only be in the middle spot in the next column. This leaves the top and bottom blocks empty, and we could fill them with $(k$ - $2)(k$ - $3) = k^2$ - $5k$ + 6 possible arrangements of different colours. However, if 'b' remained the same, it could be in the top or bottom grids, and the 2 leftover grids could be filled with $(k$ - $2)(k$ - $3) = k^2$ - $5k + 6$ possible arrangements of 2 colours. Hence, since there are 2 possible places where 'b' could be, the total number of possibilities is $2(k^2$ - $5k + 6) = 2k^2$ - $10k + 12$ possibilities. In total, there are k^2 - $5k + 6 + 2k^2$ - $10k + 12 = 3k^2$ - $15k + 18$ total possibilities for this case.

Case 3

one colour in the next column is different

As one colour is different,that means that all the colours used in the previous column (a and b) must be used. There are only two placements of a and b, as shown below.

$$\begin{array}{|c|}\hline a \\ \hline b \\ \hline \\ \hline\end{array} \text{ or } \begin{array}{|c|}\hline \\ \hline a \\ \hline b \\ \hline\end{array}$$

The block leftover in each case could be coloured in $(k$ - $2)$ ways, giving a total of $2(k$ - $2) = 2k$ - 4 ways.

Summing the three cases up, we get k^3 - $9k^2$ + $26k$ - 24 + $3k^2$ - $15k$ + 18 + $2k$ - 4 = k^3 - $6k^2$ + $13k$ - 10 possibilities in total.

$t_2 \rightarrow t_2$

$$\begin{array}{|c|}\hline a \\ \hline b \\ \hline a \\ \hline \end{array} \rightarrow \begin{array}{|c|}\hline a \\ \hline b \\ \hline a \\ \hline \end{array}$$

By observation, the bottom row can be ignored as it is of the same colouring scheme as the top row. Hence, this one can just be looked at as a 2 × n grid. As explained above when figuring out the formula for a 2 × n grid, there are k^2 - $3k$ + 3 possible colourings for the second column.

Hence, from the four formulas, we now know that:
$m_{\mathrm{n+1}}$= (k^3 - $6k^2$ + $14k$ - 13)m_{n} + (k^3 - $6k^2$ + $13k$ -10)x_{n}
$x_{\mathrm{n+1}}$= (k^2 - $3k$ + 3)x_{n} + (k^2 - $4k$ + 5)m_{n}.

To put them in a matrix, we correspond the values for $t_1 \rightarrow t_1$ and $t_1 \rightarrow t_2$ with $k(k$ - 1)(k - 2), since at the very start there are $k(k$ - 1)(k - 2) possible arrangements of t_1 columns(the first column on the most left). Similarly, we correspond the values for $t_2 \rightarrow t_1$ and $t_2 \rightarrow t_2$ with $k(k$ - 1), since that is the number of possible t_2 arrangements for the first column. We raise the square matrix (the matrix with the values we've calculated) to the (n-1)th power, because we repeat the multiplication of this matrix (n-1) times (excluding the first column). This becomes,

$$\begin{bmatrix} k^3-6k^2+14k-13 & k^3-6k^2+13k-10 \\ k^2-4k+5 & k^2-3k+3 \end{bmatrix}^{n-1} \begin{bmatrix} k(k-1)(k-2) \\ k(k-1)) \end{bmatrix}$$

To get the total number of possible solutions, the resulting 1×2 matrix will have two numbers, which need to be added together to reach the total number of solutions (the m_{n} and x_{n}).

This method can be applied to all other m$\times$n grids, to find the number of colourings for a $m \times n$ grid. If $m = 4$, there are 5 possible types of ways a column could appear.

If we write them out from top to bottom with letters:
a-b-a-b
a-b-a-c
a-b-c-a
a-b-c-b
a-b-c-d

Then we would need to work out the number of colourings for each type of column following another type. As m gets larger, this also gets larger. For $m = 4$, we would need to calculate 25 values, however some could be retained from $m = 3$, like when $t_2 \rightarrow t_2$ was the same as the number of possibilities for the next column of a $2 \times n$ grid.

The General Solution - Summary

From this, the general method should be quite clear. For any $m \times n$ grid, we find the possible ways a singular column could appear, and then calculate the number of ways any 'type' of column could follow all other types. After doing this, we map them out in a square matrix, and correspond the values of each *'type'* following another with the possible ways 1 column could appear in any type (to show the column at the start). The matrix showing the number of ways different 'types' could be at the start should have only 1 column, generally in the form of k(k - 1)...(k - i) depending on the number of different colours used. This matrix should follow the square matrix, as when matrices have to be multiplied together, the first matrix should have the same number of columns as the second matrices rows. Then, we raise the square matrix with the values we have calculated to the power of the number of columns taking away 1, because the starting column (the left-most column in the grid) has already been calculated in the matrix with a single column. The matrices would look something like the matrices listed out below.

$$\begin{bmatrix} t_1 \to t_1 & t_2 \to t_1 & ... & t_n \to t_1 \\ t_1 \to t_2 & ... & ... & t_n \to t_2 \\ ... & t_2 \to t_{n-1} & ... & ... \\ t_1 \to t_n & t_2 \to t_n & ... & t_n \to t_n \end{bmatrix}^{n-1} \begin{bmatrix} (total)t_1 \\ ... \\ ... \\ (total)t_n \end{bmatrix}$$

Finally, we calculate all of the possibilities we have listed out altogether, and add up the entries of the matrix we finally get, to reach the final total number of possible colourings of a grid. Also, m does not always have to be used to calculate the number of ways a column could appear in a $m\times n$ grid if that's how it's placed. It is always better to use n if $n>m$.

However, since this could become quite a tedious process if numbers become quite large, so a program on your computer or device could be used to reach the answer quicker. Details of the program are found on the following pages.

2. Program and Results

The purpose of the program is to check the number of combinations and visually display the results. It contains one source code file named 'question_2.py'. The program was developed using PyCharm as Development Environment.

2.1 Program Detail

The program was developed using Python version 3.7 within the PyCharm development environment. It consists of one main program, which calls 2 functions. It was written follow brute force strategy with python libraries. The main functions from libraries used to achieve the result are 'product' function to obtain Cartesian product and 'array' function to help generate and find all combinations.

First, the program will ask user for the values of row, column, and k. Then it will pass this value to the main program for execution and hence produce the result.

2.2 Program Source Code

There is only one program called "question_2.py" and it consists of the main program, and two functions called "total_possibilities" and "display_results". The purpose of "total_possibilities" function is to find all the possible combinations for the specified n, m and k values. "displa y_results" is used to display the results in a grid. The main program is used to ask for input of n, m and k, and to display the results.

Refer to the Appendix for the program source code.

2.3 Test Cases and Results

There are total of 10 cases were carried out. Since the number of test cases can go on indefinitely, only a number amount of cases have been carried out to ensure that the calculation is correct. The table below shows each test case result from the program.

Test Case	**m**	**n**	**k**	**Result**
1	1	3	4	36
2	4	1	3	24
3	2	2	2	2
4	2	2	4	84
5	2	2	5	260
6	2	2	6	630
7	2	3	2	2
8	2	3	3	54
9	3	3	3	246
10	3	2	6	13230

Bibliography

How to insert PDF into LaTeX – PDFConverters Official Website. 2020. How to insert PDF into LaTeX – PDFConverters Official Website. [ONLINE] Available at: https://www.pdfconverters.net/how-to/insert-pdf-to-latex/. [Accessed 23 July 2020].

itertools — Functions creating iterators for efficient looping — Python 3.8.5 documentation. 2020. itertools — Functions creating iterators for efficient looping — Python 3.8.5 documentation. [ONLINE] Available at: https://docs.python.org/3/library/itertools.html. [Accessed 23 July 2020].

Mathematics Stack Exchange. 2020. linear algebra - Converting recursive equations into matrices - Mathematics Stack Exchange. [ONLINE] Available at: https://math.stackexchange.com/questions/858268/converting-recursive-equations-into-matrices. [Accessed 25 July 2020].
Mathematics Stack Exchange. 2020. combinatorics - What's the name of a permutation where repetition is possible? - Mathematics Stack Exchange. [ONLINE] Available at: https://math.stackexchange.com/questions/2550416 /whats-the-name-of-a-permutation-where-repetition-is-possible. [Accessed 23 July 2020].

numpy.vstack — NumPy v1.19 Manual. 2020. numpy.vstack — NumPy v1.19 Manual. [ONLINE] Available at: https://numpy.org/doc/stable/reference/generated/numpy.vstack.html. [Accessed 23 July 2020].

Online LaTeX Equation Editor - create, integrate and download. 2020. Online LaTeX Equation Editor - create, integrate and download. [ONLINE] Available at: https://www.codecogs.com/latex/eqneditor.php. [Accessed 25 July 2020].

Python Numpy Tutorial (with Jupyter and Colab). 2020. Python Numpy Tutorial (with Jupyter and Colab). [ONLINE] Available at: https://cs231n.github.io/python-numpy-tutorial/. [Accessed 23 July 2020].

Stack Overflow. 2020. python - Get the cartesian product of a series of lists? - Stack Overflow. [ONLINE] Available at: https://stackoverflow.com/questions/533905/get-the-cartesian-product-of-a-series-of-lists. [Accessed 23 July 2020].

Steve Sque - Symbols in LaTeX and HTML. 2020. Steve Sque - Symbols in LaTeX and HTML. [ONLINE] Available at: https://www.stevesque.com/symbols/. [Accessed 25 July 2020].

Appendix

Program Source Code

The program source code is on the next page.

```python
# ------------------------------------------------------------------------------------ #
#                            Presbyterian Ladies' College                              #
#                              Mentor: Dr. David Treeby                                #
#                                 Liah Wu, Yunaa Tae                                   #
# ------------------------------------------------------------------------------------ #
from itertools import product
import numpy as np


# ------------------------------------------------------------------------------------ #
#                           Find Total Possibilities Function                          #
# ------------------------------------------------------------------------------------ #


def total_possibilities(n, m, k):
    # set colours
    letter_num = ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q"
                  "u", "v", "w", "x", "y", "z"]

    # set lists
    poss_colours = []
    row_possibilities = []
    result_list = []

    # set possible colours for combinations
    for i in range(k):
        poss_colours.append(letter_num[i])

    # when m > 1
    if m == 1 and n == 1:
        # print answer for m = 1
        for colour in poss_colours:
            result_list.append([colour])

    else:
        # find all combinations for a row
        combinations = list(product(poss_colours, repeat=n))

        # eliminate options that do not fit with question
        left = 0
        right = 1
        for option in combinations:
            allow = True
            while right < n:
                if option[left] == option[right]:
                    allow = False
                    break
                left += 1
                right += 1
            if allow:
                row_possibilities.append(option)
            left = 0
            right = 1

        # Find allowable tilings
        valid = True
        final_list = list(product(row_possibilities, repeat=m))
        total_final_list = len(final_list)
        current_group = 0
        if m == 1:
            result_list = final_list
        else:
            for next_group in range(total_final_list):
                for row in range(m):
                    for col in range(n):
                        # array of previous row
                        case_array = np.array(final_list[next_group])
                        # swaps vertical to become horizontal
                        prepare_array = case_array[:, col]
                        case_list = prepare_array.tolist()
                        no_of_cases = len(case_list)
                        for j in range(no_of_cases):
                            if j < no_of_cases - 1:
                                # check if the colours are the same or not (horizontal in the list)
                                if case_list[j] == case_list[j + 1]:
                                    valid = False
                                    break
                if current_group < next_group and valid:
                    result_list.append(final_list[next_group])
                current_group = next_group
                valid = True
    return result_list

# ------------------------------------------------------------------------------------ #
#                                Display Results Function                               #
# ------------------------------------------------------------------------------------ #


def display_results(possibilities):
    length = len(possibilities)
    print(f"Total: {length}")
    for i in range(length):
        write_result = str(np.vstack(possibilities[i])).replace("[", "").replace("]", "").replace("
        print(f"{write_result}\n")


# ------------------------------------------------------------------------------------ #
#                                       Main Program                                    #
#                             Ask for input and print out answer                        #
# ------------------------------------------------------------------------------------ #

if __name__ == "__main__":
    # ask for input for m, n and k
    print("------------INPUT------------")
    print("Please enter n, m and k, where n, m > 0, and k >= 2")
    n = int(input("n: "))
    while n < 1:
        n = int(input("n: "))
    m = int(input("m: "))
    while m < 1:
        m = int(input("m: "))
    k = int(input("k: "))
    while k < 2:
        k = int(input("k: "))
    print()
    print("------------OUTPUT------------")
    # print output
    display_results(total_possibilities(n, m, k))
```