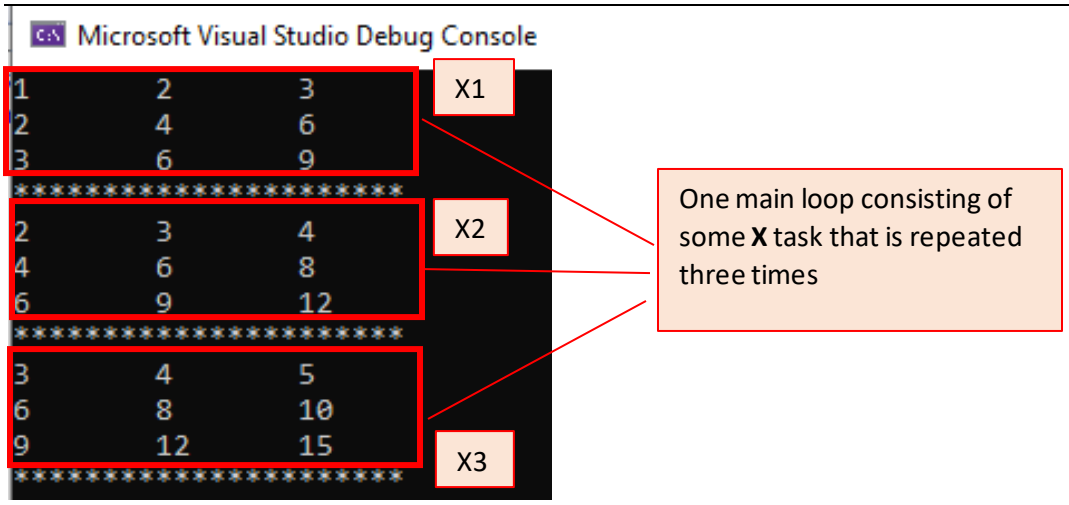
Hint for pattern



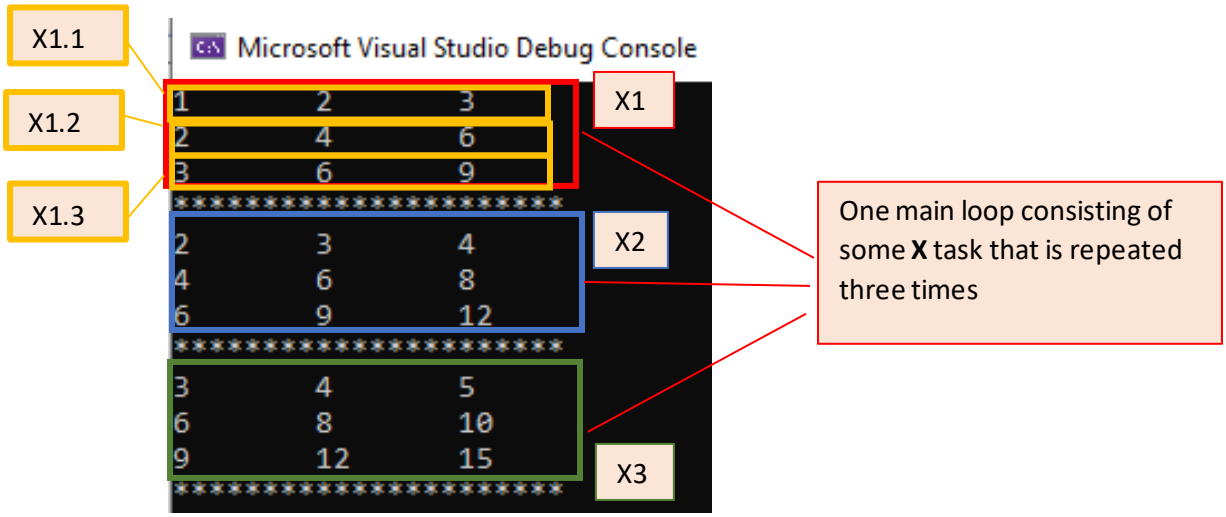
Pattern			Dry run / analysis of pattern		
1	2	3	(1+0) x 1	(2+0) x 1	(3+0) x 1
2	4	6	(1+0) x 2	(2+0) x 2	(3+0) x 2
3	6	9	(1+0) x 3	(2+0) x 3	(3+0) x 3
*********	*********	*********	*********	*********	*********
2	3	4	(1+1) x 1	(2+1) x 1	(3+1) x 1
4	6	8	(1+1) x 2	(2+1) x 2	(3+1) x 2
6	9	12	(1+1) x 3	(2+1) x 3	(3+1) x 3
*********	*********	*********	*********	*********	*********
3	4	5	(1+2) x 1	(2+2) x 1	(3+2) x 1
6	8	10	(1+2) x 2	(2+2) x 2	(3+2) x 2
9	12	15	(1+2) x 3	(2+2) x 3	(3+2) x 3

So you can have one loop that goes from 1 to 3. Thus inside main loop we need to perform the tasks x1, x2 and x3

```
For (i=1 to 3)
{
        //perform the tasks x1
        //perform the tasks x2
        //perform the tasks x3
}
```

Now lets explore the X task (X1 , X2, X3) that has to be repeated 3 times. Each sub task further consists of drawing a 3 by 3 pattern. So we can break down these tasks to smaller sub tasks:

Breaking down X1 into smaller sub tasks:



These subtasks should be nested with out main loop. We can write sub task X1 as:

```
For (i=1 to 3)
{
        //perform the tasks x1
                For (j=1 to 3) Perform x1.1,  x1.2,  x1.3 tasks
                {


                }
        //perform the tasks x2
                For (j=1 to 3) Perform x2.1,  x2.2,  x2.3 tasks
                {


                }
        //perform the tasks x3
                For (j=1 to 3) Perform x3.1,  x3.2,  x3.3 tasks
                {


                }
```

}

If we further sub divide our tasks and analyze the pattern, we can easily see that the three loops above can be replaced with one nested loop (with loop variable as “k”).

```
For (i=1 to 3)
{
        For (j=1 to 3)
        {
                For (k=1 to 3)
                {
                        //insert the logic or formula that can print the required
                        pattern
                }

        }
}
```

Dry run the and execute these loops and see what are the values of loop variables I, j and k. This will help in deciding which values should be added or multiplied together to generate the pattern. You can run the script like the following to see what are the values of loop variable so that you can use them accordingly.

```
#include <iostream>
using namespace std;
int main()
{
        int i, j,k;

        for (i = 1; i <= 5; i++)
        {
                cout << "Value of i:";
                cout<< i << endl;
                cout << "**********************";
                cout << endl;
                for (j = 1; j <= 5; j++)
                {
                        cout << "Value of j:";
                        cout << j << endl;
                        cout << "**********************";
                        cout << endl;
                        for (k = 1; k <= 5; k++)
                        {
                                cout << "Value of k:";
                                cout << k << endl;
                        }
```

```
                cout << "**********************";
                cout << endl;

            }
        }

        return 0;
}
```