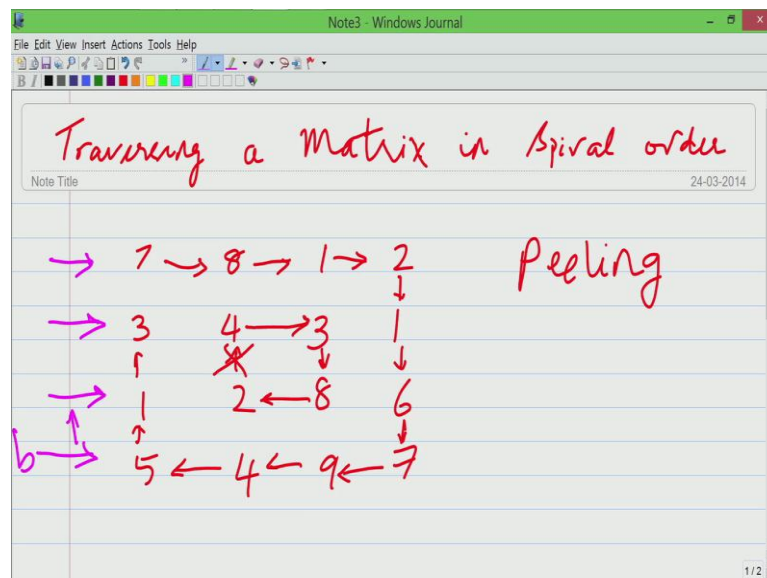


**Programming Data Structures, Algorithms**  
**Prof. Shankar Balachandran**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Madras**

**Lecture - 16**  
**Solution: Second Week Programming Assignment**

Hello, welcome students. So, in this session solving problems, I thought I will show you the solution to this problem on printing the matrix in a spiral. So, given a matrix and we have to traverse it in a spiral order and print the elements. So, bear with me for today, I have a terrible sore throat.

(Refer Slide Time: 00:30)



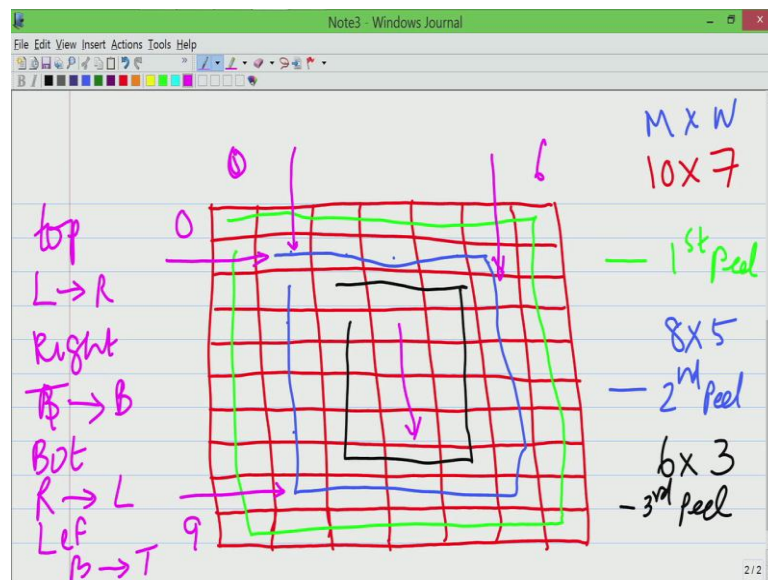
So, let us say here from the Horses mouth, today you are going to here from the Horse itself. So, let us look at this matrix here, it is a 4 cross 4 matrix which is given as a public test case for you. And when we say traverse this in spiral manner, what we going to do is we want to go in this order. So, we want to traverse the top row first and after the top row, we traverse the right most column, we traverse the bottom most row and traverse the left most column.

At this point, we are done with all the outer layer and now we are supposed to go to the inner layer and do things inside and so, on. So, I am going to define something called peeling. So, think of it as what you do with an onion. So, you take an onion and you remove one layer completely, what you are done with is, you have removed one layer and you still left with an onion which is similar to what you had earlier. Just like that, if

you take the elements from the top row, then strip of elements in the right, strip of elements in the bottom and strip of elements on the left.

If you start with a rectangle, you would be left with another rectangle. In this case, you would be left with a rectangle 4, 3, 2 and 8, if you remove the outer once. So, you peel one layer and you are left with another rectangle and on this rectangle, again I can start at the left top and move one step at a time and print all these elements. So, now I am done with one layer. So, of course, this arrow must not be there, I am done with another layer inside and so, on. So, if I keep peeling one layer after the another, I will be left with the sub problem which is similar to the original problem itself. So, this is a nice structure to this whole setup. So, now the question is, how long do I keep peeling and what are the corner cases?

(Refer Slide Time: 02:31)



So, to show that I have a small grid, which is 10 rows cross 7 columns here, I want to see, how many times we can actually do this operation of going on the top, traversing on the right, being in the bottom and then again traversing on the left. How many times we can do that? So, I just follow my marker color carefully. So, I start with the left most cell on the top, I do one traversal. Then, in some sense I turn right, then turn right and then go all the way here.

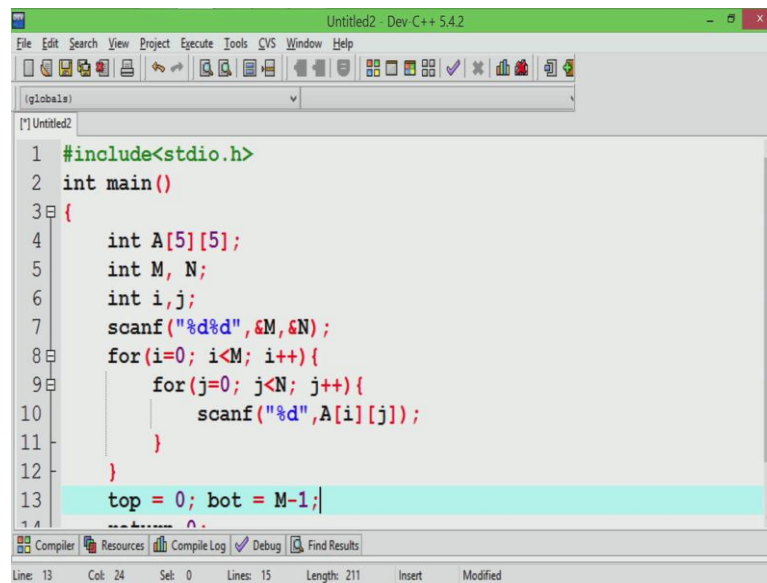
So, that takes care of one layer of peel. So, that is my green line here is the first peel. Once I am done with that I am left with a sub problem, which is not a 10 cross 7 anymore. So, let us go and count what we have. So, we have 1, 2, 3, 4, 5, 6, 7, 8 rows

and 1, 2, 3, 4, 5 columns. So, this is not a surprise, because we started with some  $m$  cross  $n$  which is 10 cross 7 and from there we remove 2 rows and 2 columns. So, now we start with a 8 cross 5.

So, now let us peel the outer most layer of the 8 cross 5. So, that is the second peel, once is we remove it, as you can expect, we will have 6 rows and 3 columns. So, there will be the third layer. So, let us look at the third layer, we will have 6 rows and 3 columns. So, and then again I can peel once more. So, that is the third peel and once the third peel is over, now it looks like I have a corner case. So, I do not have a proper rectangle anymore, it is not 2 rows or 2 columns anymore. In fact, I have just 1 row here. So, I will mark that with this color. So, I have just 1 column here that I need to traverse. So, in all the other once, I can start at the left top of the rectangle and move right towards the right here and then move bottom, move left and move up. So, essentially what we are doing is, on the top most row, we traverse from left to right. Then, on the right most column, it traverse from top to bottom. On the bottom most column, you traverse from right to left and in the left most column, you traverse from bottom to top.

So, and this is the process that you keep repeating, only that what is your top, right, bottom and left keeps changing, as you go along. So, initially your top is row number 0 and the bottom is row number 9, initially, but as we go along... So, and in that case, the left would be 0 and the right would be 6, because we have a 10 cross 7 system, that will be a first layer. Once you have that your top will become. So, this is the left, after one round of peeling. This is the right, after one round of peeling. This line will be the top, after one round of peeling and this will be the bottom of after one round of peeling. So, and we are going to do this repeatedly. So, I am going to take this and see how to translate that into our program. So, to do that, I am opening up my local editor.

(Refer Slide Time: 06:13)

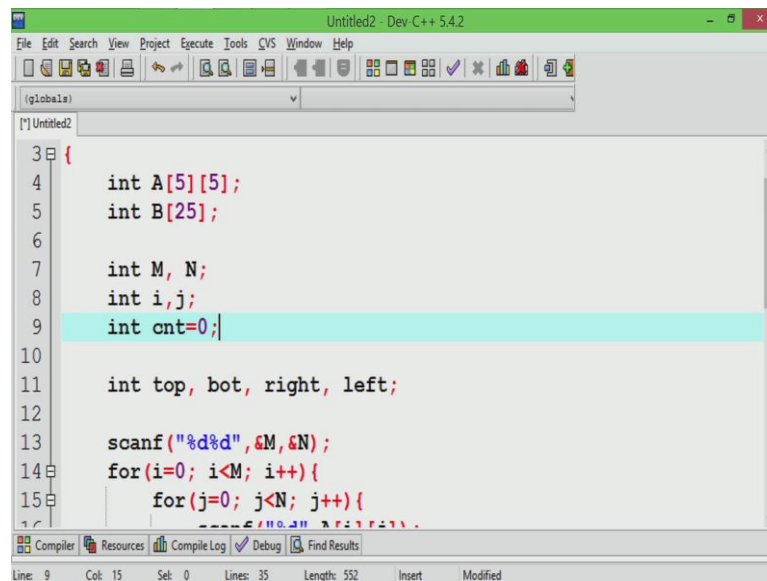


```
1 #include<stdio.h>
2 int main()
3 {
4     int A[5][5];
5     int M, N;
6     int i, j;
7     scanf("%d%d", &M, &N);
8     for(i=0; i<M; i++){
9         for(j=0; j<N; j++){
10            scanf("%d", A[i][j]);
11        }
12    }
13    top = 0; bot = M-1;
```

So, what I am going to do is as before, I will go and scan the matrix first and what are the things that I need. So, your problem specification said, you should support something for a 5 cross 5 matrix. So, I am going to declare int A of 5 comma 5 and we need to scan coordinates of the rectangle, what is the size of the rectangle. I am going to assume that rows is M and columns is N and I am going to declare two variables called int i and j, int i and j will declare other things as we go along.

So, for now what I am going to do is, for i equals 0, i less than M, i plus plus, for j equals 0, j less than N, j plus plus scanf one integer which is A of i comma j. So, before doing that I need M and N itself. So, I will do that outside, scanf ampersand M and ampersand N. So, assuming that I have read the number of rows and columns and both of them being less than equal to 5. So, when I am going to read M cross N matrix inside the system. So, I want to imitate, what I did earlier in the graphics.

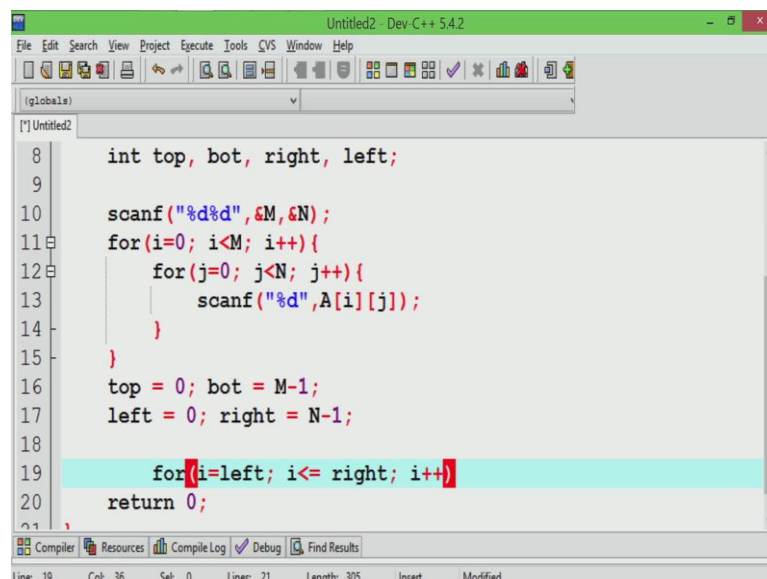
(Refer Slide Time: 07:56)



```
3 {
4     int A[5][5];
5     int B[25];
6
7     int M, N;
8     int i, j;
9     int cnt=0;
10
11     int top, bot, right, left;
12
13     scanf("%d%d", &M, &N);
14     for(i=0; i<M; i++){
15         for(j=0; j<N; j++){
```

So, I know that the top most row is actually row number 0 and bottom most row is row M minus 1. So, I have the top and bottom and I need the left and right, I will start with left most being the 0th column and right most being the N minus 1th column. So, this takes care of C style of indexing. So, only that I have been declare top, bottom and so, on. I will declare that now, in top, bottom, right and left. Now, what I am going to do. So, let us write code for one layer, peeling one layer let us write code for that.

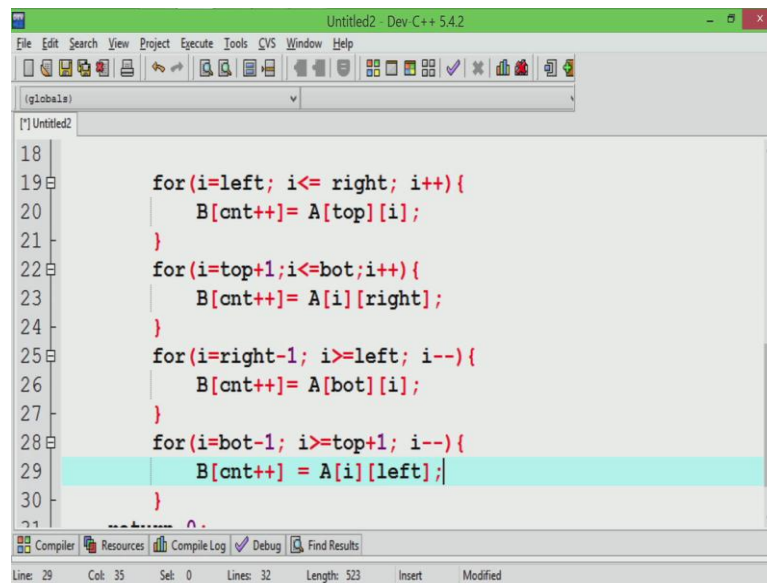
(Refer Slide Time: 08:32)



```
8     int top, bot, right, left;
9
10    scanf("%d%d", &M, &N);
11    for(i=0; i<M; i++){
12        for(j=0; j<N; j++){
13            scanf("%d", A[i][j]);
14        }
15    }
16    top = 0; bot = M-1;
17    left = 0; right = N-1;
18
19    for(i=left; i<= right; i++)
20        return 0;
```

So, what I am going to do is, I am going to start at the top and I am going to traverse from left to right so, from left to right, one element at a time.

(Refer Slide Time: 08:55)



```
18
19     for(i=left; i<= right; i++){
20         B[cnt++]= A[top][i];
21     }
22     for(i=top+1; i<=bot; i++){
23         B[cnt++]= A[i][right];
24     }
25     for(i=right-1; i>=left; i--){
26         B[cnt++]= A[bot][i];
27     }
28     for(i=bot-1; i>=top+1; i--){
29         B[cnt++] = A[i][left];
30     }
```

Let us say, if I access A of top comma i, I am going to do something to A of top comma i, but that will take care of doing everything on the top most row. Because, my row indexes starts, I keep changing it goes from top comma left in one step to top comma right. So, it goes one step after the other and what work I am going to do, I am going to show later. Then, I need to start at one row below the top most row,. So, I will start with that row below the top most row. Then, I will go up to the bottom most row, again I am increasing i.

Here what I am going to do is, I am going to access A of i which is the row number and right most column, I am going to do something on that later. So, now I am taking care of traversing on the top and traversing on the right. Then, what I am going to do is, I am going to start from right minus 1 and go up to left. But, I am going in the reverse direction. So, I am going do i minus minus and in this case, we are accessing A of bottom comma i.

And finally, i will go from bottom minus 1, then i want to go up to top plus 1, i minus minus. Again, I am going to access A of i comma left. So, you can see what is happening here. So, this part line numbers 19, 20 and 21, you are accessing A of top comma i, but i goes from left to right. Then, line numbers 22, 23 and 24 is accessing A of i comma right. So, since right is fixed, you are accessing a column and which rows are you accessing, you are accessing from top plus 1th row to the bottom most row.

And line numbers 25, 26 and 27, you are accessing the bottom most row, but you are

going from right side to left side. You are starting with right minus 1, because you have already touch the right bottom cell and you go greater than equal to left and finally, go from bottom minus 1 greater than equal to top plus 1 and i minus minus. So, the first two loops move in the left and bottom direction and the second, the bottom two loops move in the left and above direction.

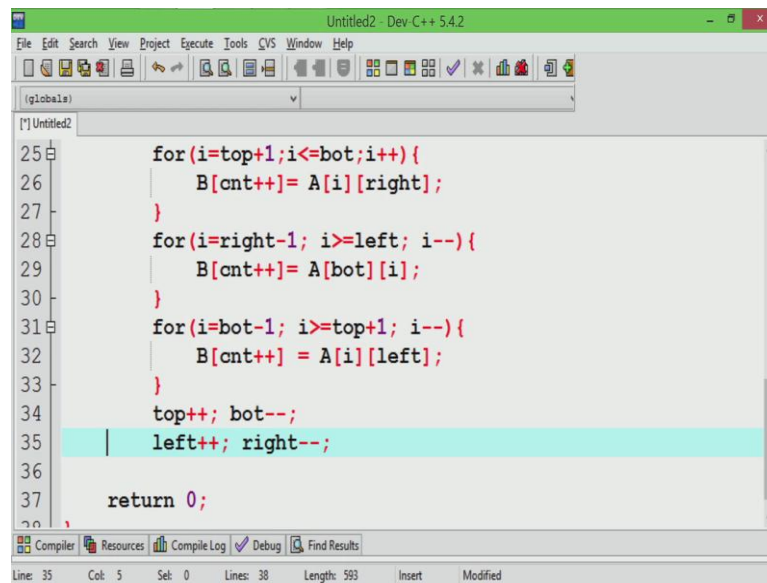
So, this is what we have. Now, we have to do a few things. So, one thing I am going to do first is, I am not going to assume that this is printed directly, because. So, many of you run into problems with presentation errors. I am going to show a technique of, how to avoid presentation errors. So, instead of actually printing right here, what I am going to do is, I am going to store in a matrix called B that I access A.

So, I am going to declare B and other things appropriately, what I am going to do is, I am going to B of cnt plus plus. So, count is a variable I am going to run and I am going to ensure that this records what I visited are not printed and I am going to do that on all of them. So, I do that on all of them. So, now all I need is, I should ensure that B is also declared properly. So, ((Refer Time: 13:37)) A is of size 25, I could not visit more than 25 elements.

So, I am going to keep B of 25 and since count is a variable that is counting the elements, I am going to put in count equals 0. So, now what I have is I have peeled exactly one layer. So, I move from left to right, top plus 1 to bottom, right minus 1 to left and bottom minus 1 to top plus 1 and at the end of this, I would have peeled exactly one layer. ((Refer Time: 14:09)) So, let us go back to this picture, let us see what we would do after peeling one layer.

So, ((Refer Time: 14:18)) in this picture after we peeled one layer, top can actually being incremented by 1, because this was the earlier top. Now, the top most row is this row and bottom can be reduced by one layer. So, this was earlier bottom and now it is going up. This will be the row which is bottom. Similarly, left moves to the right side by one column and the right can move to the left by one column,. So, I will put that in.

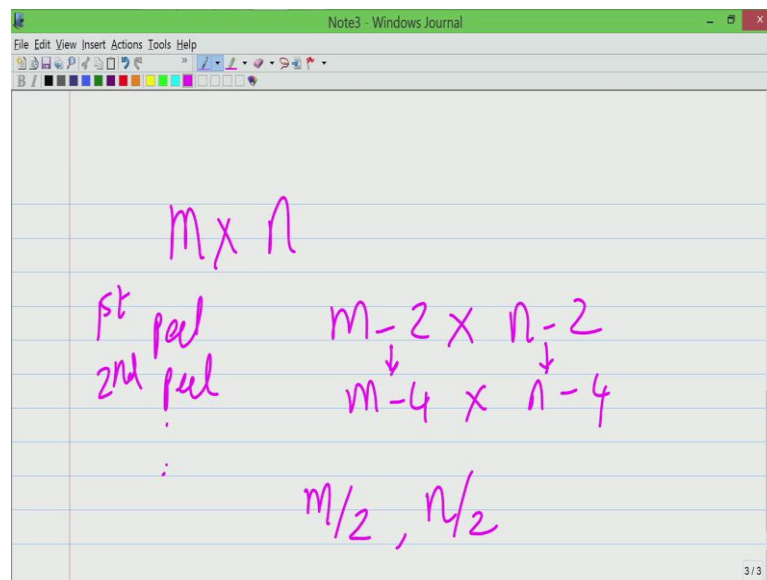
(Refer Slide Time: 14:47)



```
25     for(i=top+1;i<=bot;i++){
26         B[cnt++]= A[i][right];
27     }
28     for(i=right-1; i>=left; i--){
29         B[cnt++]= A[bot][i];
30     }
31     for(i=bot-1; i>=top+1; i--){
32         B[cnt++] = A[i][left];
33     }
34     top++; bot--;
35     left++; right--;
36
37     return 0;
```

So, top can move up, it is can move down by one step, bottom can move up by one step, left can move to the right by one step and right can move to the left by one step. Now, the second question arises, this takes care of one peeling, but we need to see how many times we can peel?

(Refer Slide Time: 15:17)



So, let us go back to the picture and if I give you a general  $m$  cross  $n$ , how many times can you really peel? So, let us see after the first peel, before the first peel you have  $m$  cross  $n$ . After the first peel, what do you get? You will have a  $m$  minus 2 cross  $n$  minus 2 rectangle system. After the second peel, you will have  $m$  minus 4 cross  $n$  minus 4, because you remove two rows from here and you remove two columns from there and

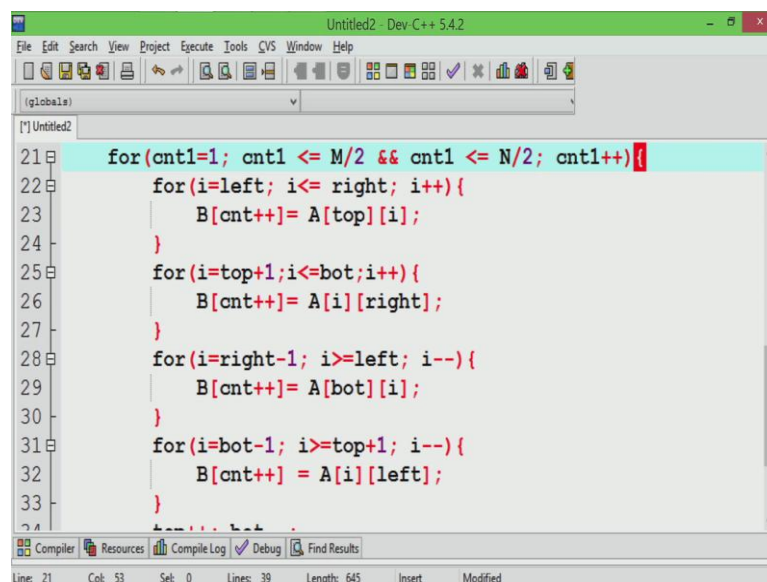


we can keep doing this, but how many times can you keep doing this?

You cannot keep do it forever, you can do it at most  $m$  by 2 times or  $n$  by 2 times, whichever is smaller. So, let me repeat this, you are removing two columns at a time and simultaneously you are removing two rows,. So, which will peel the outer layer of the rectangle and how many times can you do this? You cannot do it more than  $m$  by 2 times or  $n$  by 2 times. So, that is the first thing I am going to do.

So, I am going to ensure that this process of peeling,. So, the code that you are seeing is for peeling from line number 22 to line number 35 is, what you are seeing for peeling accept that I have not said how many times you can peel.

(Refer Slide Time: 16:36)



```
21 for(cnt1=1; cnt1 <= M/2 && cnt1 <= N/2; cnt1++){
22     for(i=left; i<= right; i++){
23         B[cnt++] = A[top][i];
24     }
25     for(i=top+1; i<=bot; i++){
26         B[cnt++] = A[i][right];
27     }
28     for(i=right-1; i>=left; i--){
29         B[cnt++] = A[bot][i];
30     }
31     for(i=bot-1; i>=top+1; i--){
32         B[cnt++] = A[i][left];
33     }
34 }
```

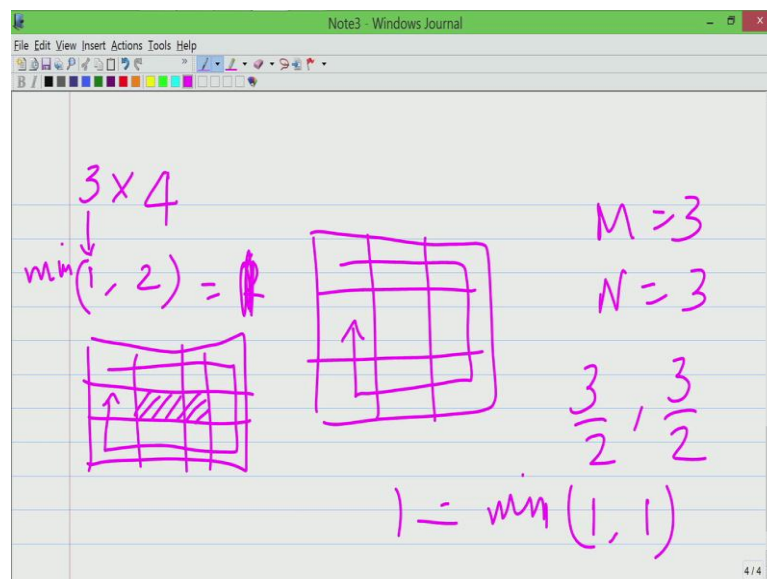
So, I am going to declare another variable called count 1 and I am going to say count 1 must be less than or equal to both  $M$  by 2 and  $N$  by 2. What this takes care of is, it ensures that peeling happens exactly either  $M$  by 2 times or  $N$  by 2 times, whichever is lesser.

(Refer Slide Time: 17:04)

```
Untitled2 - Dev-C++ 5.4.2
File Edit Search View Project Execute Tools CVS Window Help
[global]
[*] Untitled2
25 |         for(i=top+1;i<=bot;i++){
26 |             B[cnt++]= A[i][right];
27 |         }
28 |         for(i=right-1; i>=left; i--){
29 |             B[cnt++]= A[bot][i];
30 |         }
31 |         for(i=bot-1; i>=top+1; i--){
32 |             B[cnt++] = A[i][left];
33 |         }
34 |         top++; bot--;
35 |         left++; right--;
36 |     }
37 |     return 0;
Compiler Resources Compile Log Debug Find Results
Line: 36 Col: 5 Sel: 0 Lines: 38 Length: 641 Insert Modified
```

So, now this takes care of peeling various things. But, then sometimes we are left with the corner case. So, let us see one of this corner cases.

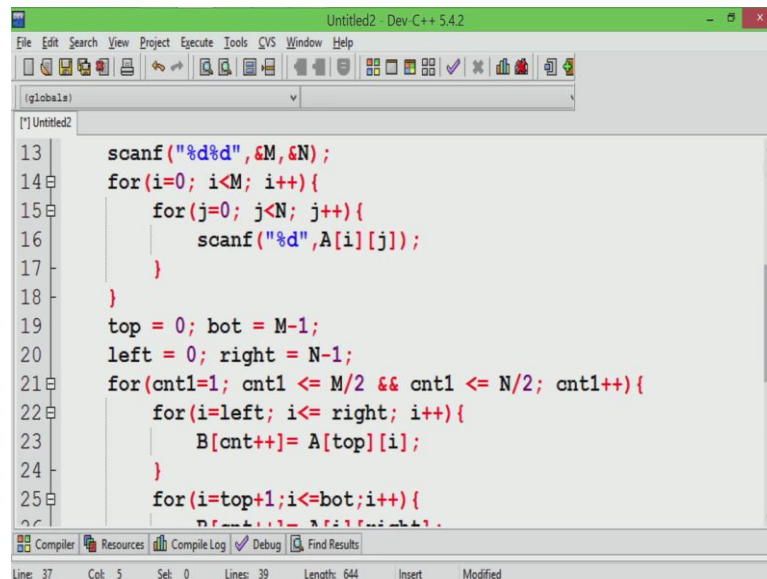
(Refer Slide Time: 17:17)



For example, favor 3 cross 3, favor 3 cross 3 system like this. So, M equals 3 and N equals 3. So, I said we can peel at most 3 by 2 times or 3 by 2 times, whichever is smaller. So, 3 by 2 happens to be 1 which is an integer division, 3 by 2 happens to be 1 also. Minimum of 1 comma 1 is 1 time. So, it can be peeled exactly 1 time. So, you do this, you can peel it exactly 1 time. So, let us take another small example which is the 3 cross 4 systems.

So, the claim is that you can either peel it, minimum of 1 time or 2 times, whichever is smaller. So, minimum of 1 comma 2 is 1 itself. So, I can peel it exactly once. Let us verify that claim,. So, I have 3 rows and 4 columns. How many times can I peel it completely? I will start from here, go up here and so, on. These two cells are part of inner one which I cannot peel. So, I peeled it once. So, in general I said we can peel it a minimum of  $M$  by 2 times or  $N$  by 2 times and that is exactly, what this checked us.

(Refer Slide Time: 18:32)



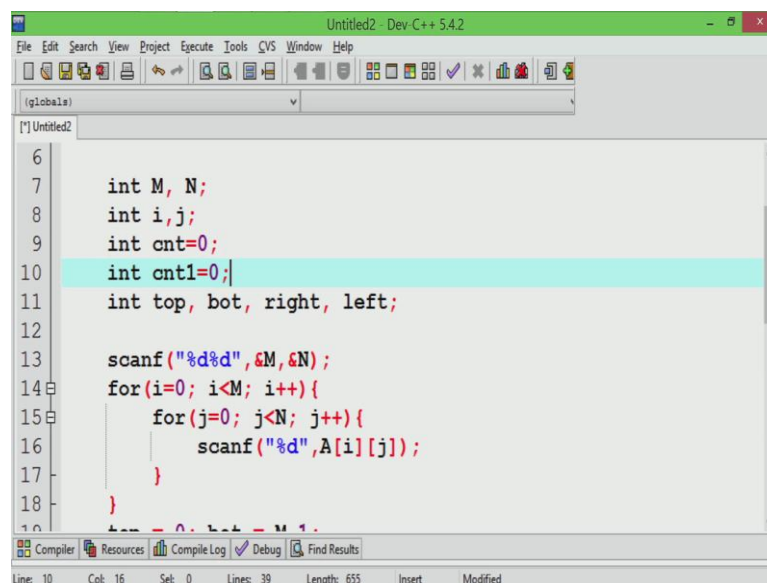
```

13 scanf("%d%d", &M, &N);
14 for(i=0; i<M; i++){
15     for(j=0; j<N; j++){
16         scanf("%d", A[i][j]);
17     }
18 }
19 top = 0; bot = M-1;
20 left = 0; right = N-1;
21 for(cnt1=1; cnt1 <= M/2 && cnt1 <= N/2; cnt1++){
22     for(i=left; i<= right; i++){
23         B[cnt1]= A[top][i];
24     }
25     for(i=top+1; i<=bot; i++){

```

So, cnt1 is less than or equal to  $M$  by 2 and cnt1 is less than or equal to  $N$  by 2. So, of course, I need to declare cnt1 itself.

(Refer Slide Time: 18:40)



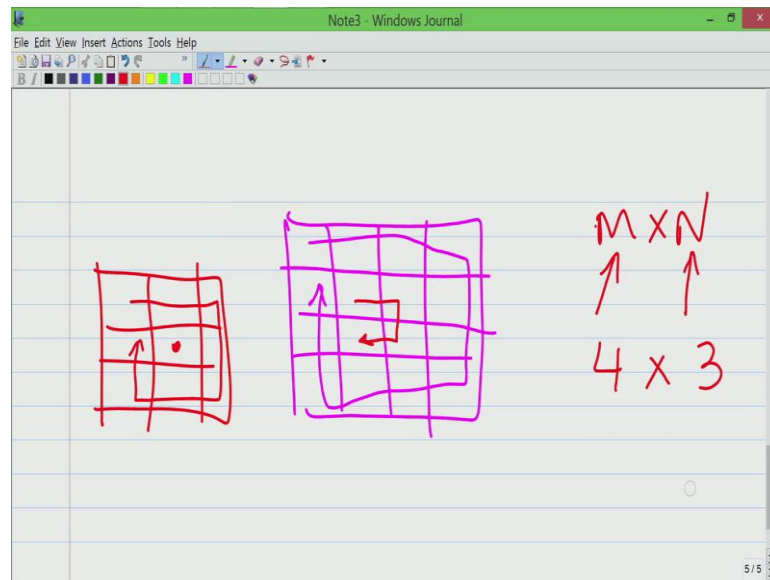
```

6
7 int M, N;
8 int i, j;
9 int cnt=0;
10 int cnt1=0;
11 int top, bot, right, left;
12
13 scanf("%d%d", &M, &N);
14 for(i=0; i<M; i++){
15     for(j=0; j<N; j++){
16         scanf("%d", A[i][j]);
17     }
18 }

```

So, I am going to declare it here, cnt1 is also equal to 0. So, now I have the various things, accept for the fact I have not taken care of what the corner cases, the final case. So, before I do that let me take one more example.

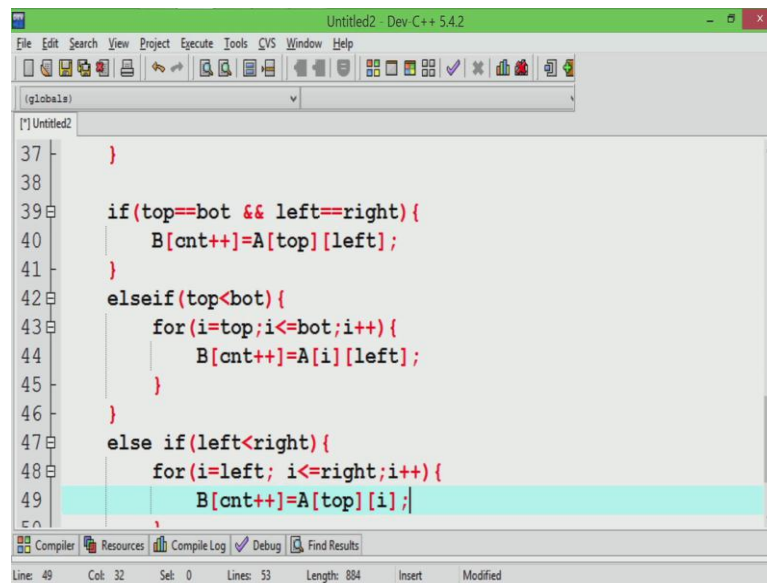
(Refer Slide Time: 18:57)



Let me take a 4 cross 4 system. So, this is the 4 cross 4 system, if I start peeling, I will start from here and end up here with one peel and the second peel would start from here and end up here. In this case, it does not leave anything to be traversed at all, actually it traverses everything. So, in general if you have a M cross N which are both even, what would happen is, you can actually do peels which is minimum of M by 2 times or N by 2 times, you will actually touch all the elements.

But, if one of them is odd, let us say M is odd and N is even for example, 4 cross 3, then you would peel, but you would be left with either 1 row or 1 column. So, we saw that example here. In a 3 cross 4 system, it is 3 rows and 4 columns, 3 is odd we can do one peeling. But, then we will be left with 1 row. But, if you do a 4 cross 3 system, you will be left with 1 column and if we start with something like a 3 cross 3 system, where both are odd, both M and N are odd, you will traverse the exactly once and then you will be left with one element in between.

(Refer Slide Time: 20:21)



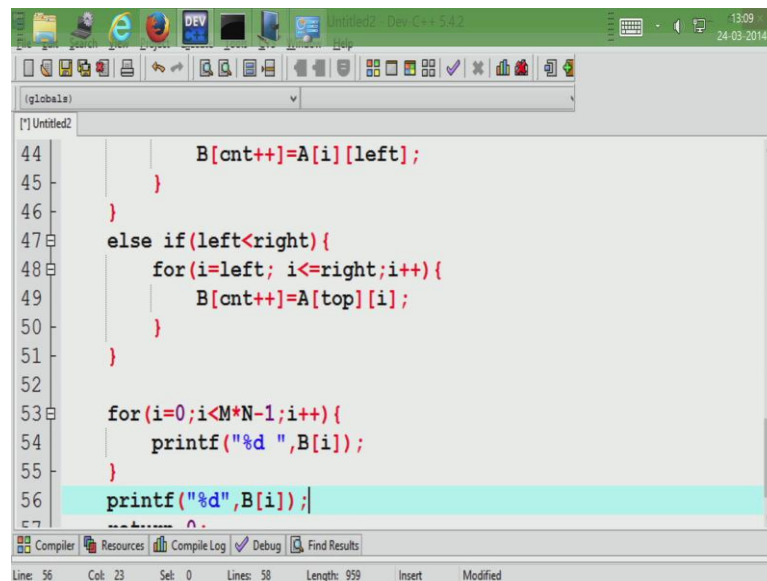
```
37     }
38
39     if(top==bot && left==right){
40         B[cnt++] = A[top][left];
41     }
42     elseif(top<bot){
43         for(i=top; i<=bot; i++){
44             B[cnt++] = A[i][left];
45         }
46     }
47     else if(left<right){
48         for(i=left; i<=right; i++){
49             B[cnt++] = A[top][i];
```

So, I am going to use this idea and write the rest of the program. So, at the end of it, if you are done with everything and let us say, both your top and bottom co inside and left and right co inside, it means that you are left with exactly one element. So, I am going to do, I am going to traverse that,. So, A of left. So, what this does is, it takes care of touching that exactly one element that you left out. Else, let us say top is still less than bottom, it means you have a column that you have to traverse from top to bottom, but it is just a one single column and which is that column, it is either left or right.

In fact, left and right should both be same at this point of time. So, what I am going to do is, I am going to traverse from the top up to the bottom and what is the element that I am going to access, I am going to access A of row which is i and column which is either left or right, it does not matter. At this point, actually left should be equal to right. Else, if we get this condition that left is actually less than right, which means they have not coincided at yet.

Now, you are actually ready to go from left most column to the right most column, but it is just one single row and which row is that, it is either the bottom or the top row. In fact, both should be the same at this point of time. So, I am going to traverse that A of, I am going to take top, but even if you put bottom here, it is not wrong, it do that and this point at the end of line 51 now, the whole traversal is over.

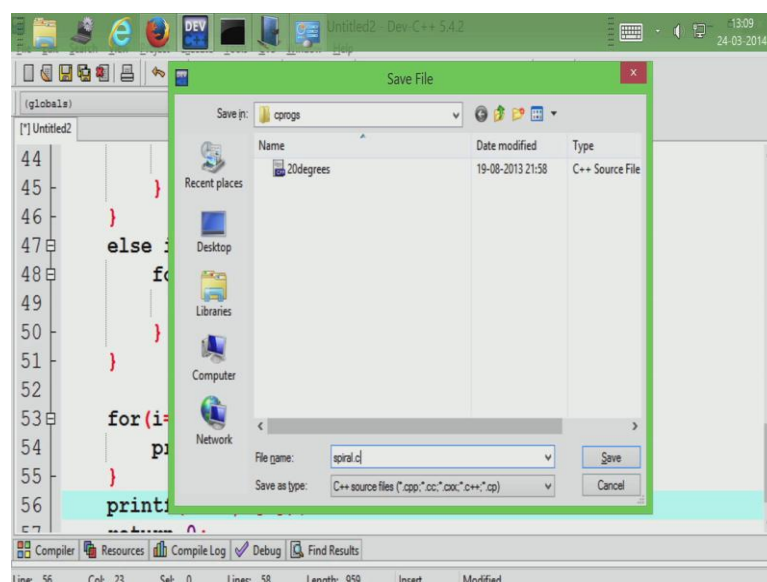
(Refer Slide Time: 22:22)



```
44     B[cnt++]=A[i][left];
45     }
46 }
47 else if(left<right){
48     for(i=left; i<=right;i++){
49         B[cnt++]=A[top][i];
50     }
51 }
52
53 for(i=0;i<M*N-1;i++){
54     printf("%d ",B[i]);
55 }
56 printf("%d",B[i]);
```

So, final thing that we need to do is, just we able to print. So, what I am going to do is, I am going to start from  $i$  equals 0 which is the 0th element of B and I am going to traverse overall. There are  $M$  times  $N$  minus 1 elements, I am going to print all of them with the space,. So, percentage d B of i. So, this takes care of printing  $M$  cross  $N$  elements for the last element, we do not want a space. So, I will just printed directly and this is my whole program.

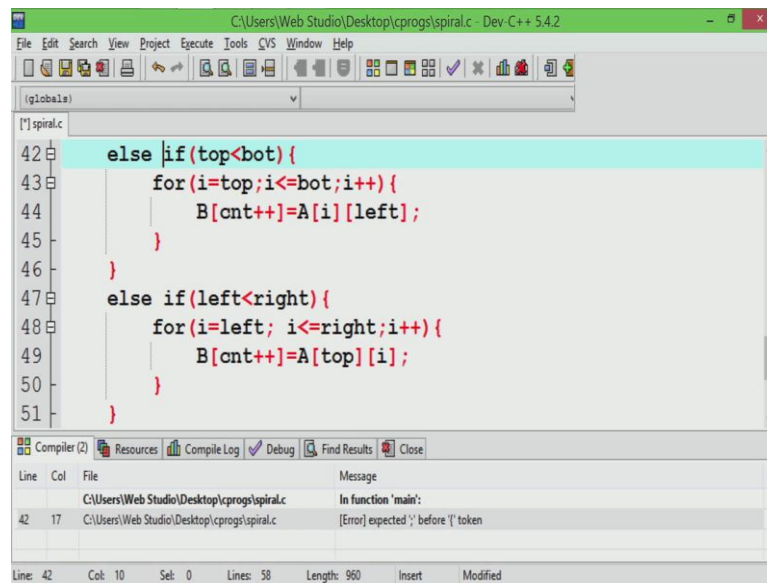
(Refer Slide Time: 23:07)



```
44
45
46 }
47 else :
48 fo
49
50 }
51 }
52
53 for (i=
54 pr
55 }
56 print
```

So, I am going to save this as spiral dot c. Let me compile it.

(Refer Slide Time: 23:14)



```
42 else if (top < bot) {
43     for (i = top; i <= bot; i++) {
44         B[cnt++] = A[i][left];
45     }
46 }
47 else if (left < right) {
48     for (i = left; i <= right; i++) {
49         B[cnt++] = A[top][i];
50     }
51 }
```

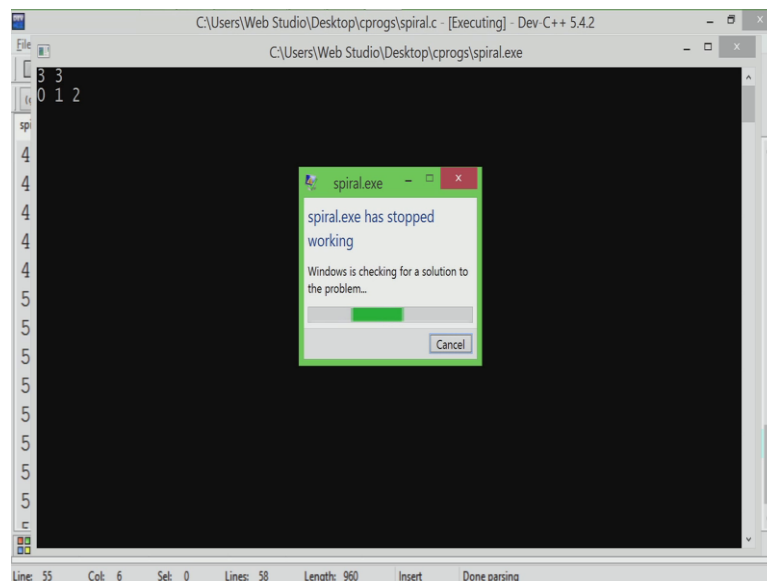
Compiler (2) Resources Compile Log Debug Find Results Close

Line	Col	File	Message
42	17	C:\Users\Web Studio\Desktop\cprogs\spiral.c	In function 'main':
		C:\Users\Web Studio\Desktop\cprogs\spiral.c	[Error] expected ';' before '[' token

Line: 42 Col: 10 Sel: 0 Lines: 58 Length: 960 Insert Modified

So, there is a small problem here. So, this time there is no compilation error.

(Refer Slide Time: 23:24)



```
3 3
0 1 2
```

spiral.exe has stopped working

Windows is checking for a solution to the problem...

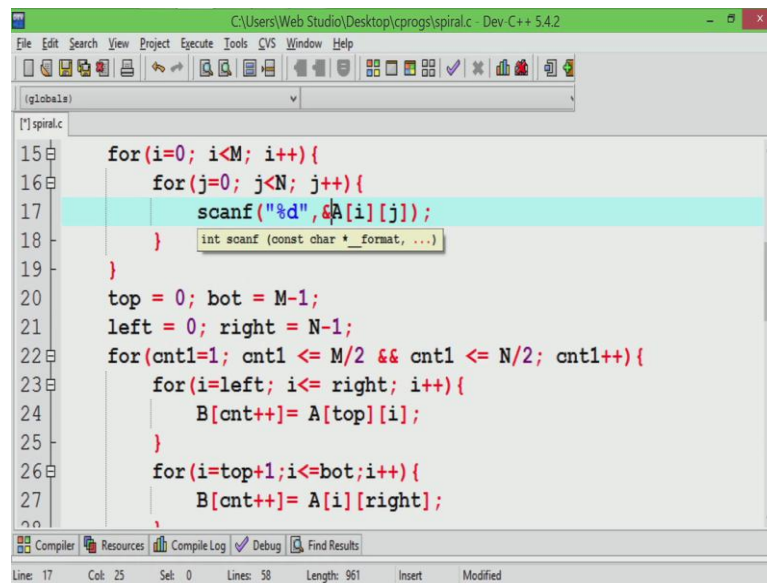
Cancel

Line: 55 Col: 6 Sel: 0 Lines: 58 Length: 960 Insert Done parsing

Let me run it, I am going to test it on some of the public test cases. The first public test case is this, this 3 comma 3 system and it has 0, 1, 2 and so, on. So, there is a small problem, I think I forgot to put a scanf with ampersand. Let me go and check.



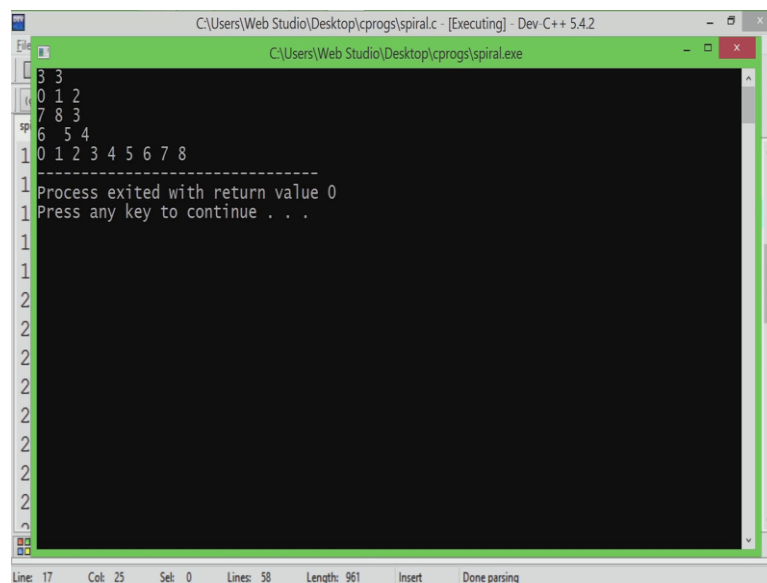
(Refer Slide Time: 23:45)



```
15 for(i=0; i<M; i++){
16     for(j=0; j<N; j++){
17         scanf("%d", &A[i][j]);
18     }
19 }
20 top = 0; bot = M-1;
21 left = 0; right = N-1;
22 for(cnt1=1; cnt1 <= M/2 && cnt1 <= N/2; cnt1++){
23     for(i=left; i<= right; i++){
24         B[cnt1++] = A[top][i];
25     }
26     for(i=top+1; i<=bot; i++){
27         B[cnt1++] = A[i][right];
```

So, this problem scanf was without the ampersand, it was a mistake. So, let me save it and compile it once more.

(Refer Slide Time: 23:51)

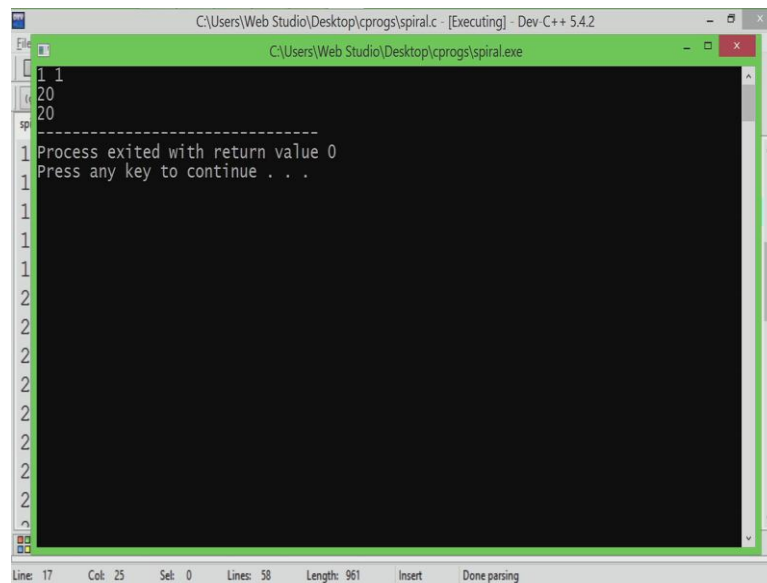


```
3 3
0 1 2
7 8 3
6 5 4
0 1 2 3 4 5 6 7 8
-----
Process exited with return value 0
Press any key to continue . . .
```

I am running it now. So, 0, 1, 2 and 7, 8, 3 and finally, 6, 5, 4 and this seems to give the output 0, 1, 2, 3, 4, 5, 6, 7, 8 which is what I expected.



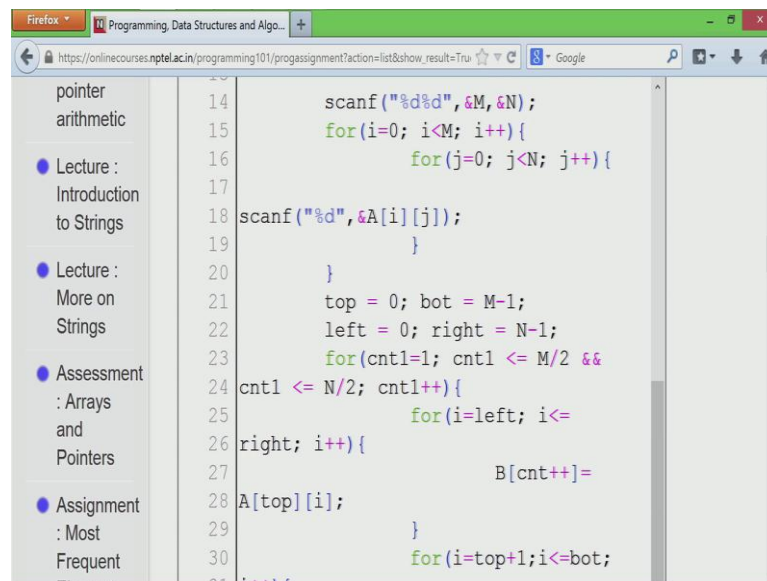
(Refer Slide Time: 24:10)



```
C:\Users\Web Studio\Desktop\cprogs\spiral.c - [Executing] - Dev-C++ 5.4.2
C:\Users\Web Studio\Desktop\cprogs\spiral.exe
1 1
20
20
-----
Process exited with return value 0
Press any key to continue . . .
```

Let us try it on some of the corner cases, what if I have is only a 1 cross 1 system and that is the last test case 1 cross 1, the output is supposed to be 20. So, this seems to be. So, let me take this and put it in the system.

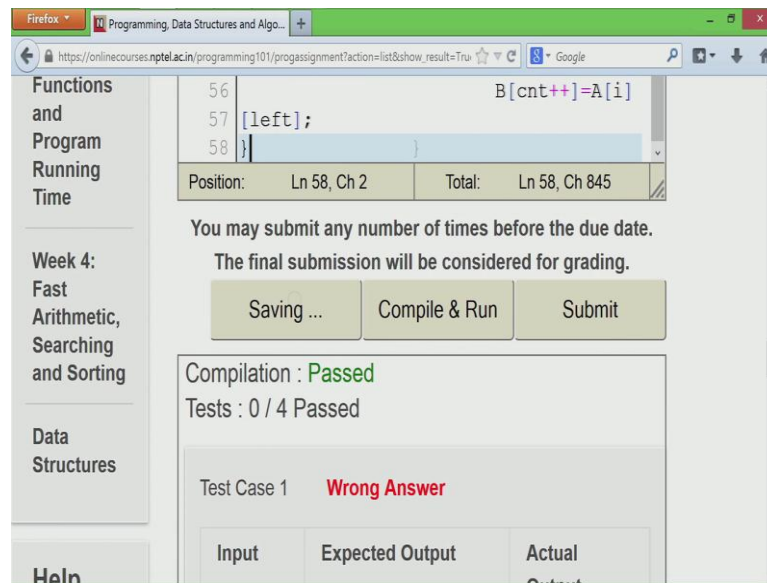
(Refer Slide Time: 24:24)



```
Firefox - Programming, Data Structures and Algo...
https://onlinecourses.nptel.ac.in/programming101/progassignment?action=list&show_result=True
scanf("%d%d", &M, &N);
for(i=0; i<M; i++){
    for(j=0; j<N; j++){
scanf("%d", &A[i][j]);
    }
}
top = 0; bot = M-1;
left = 0; right = N-1;
for(cnt1=1; cnt1 <= M/2 &&
cnt1 <= N/2; cnt1++){
    for(i=left; i<=
right; i++){
        B[cnt++] =
A[top][i];
    }
    for(i=top+1; i<=bot;
```

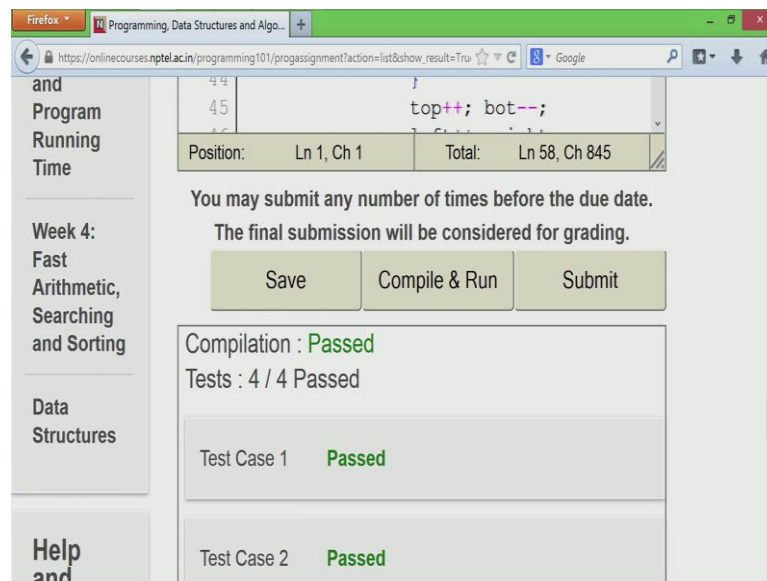
So, as before I have this saved in my setup like last week. So, I have opened it up, I just copied and pasted in it.

(Refer Slide Time: 24:34)



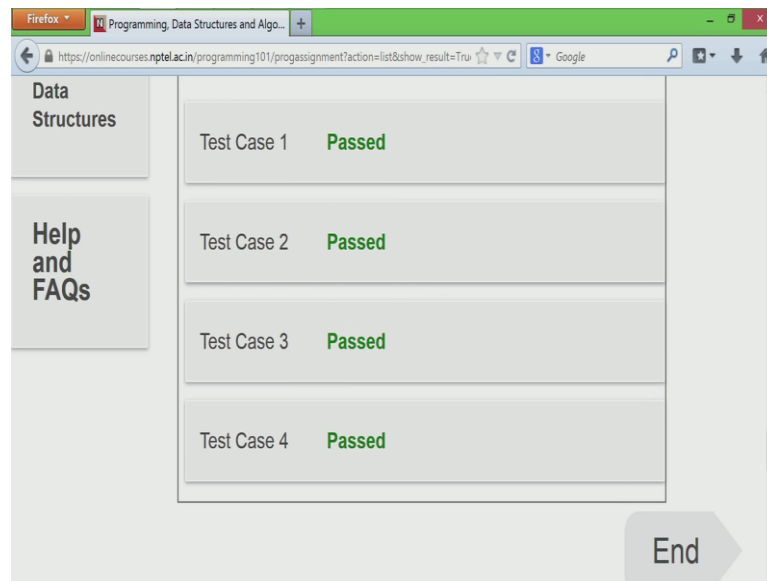
So, I will save and compile and run. So, let me see hopefully this is.

(Refer Slide Time: 24:50)



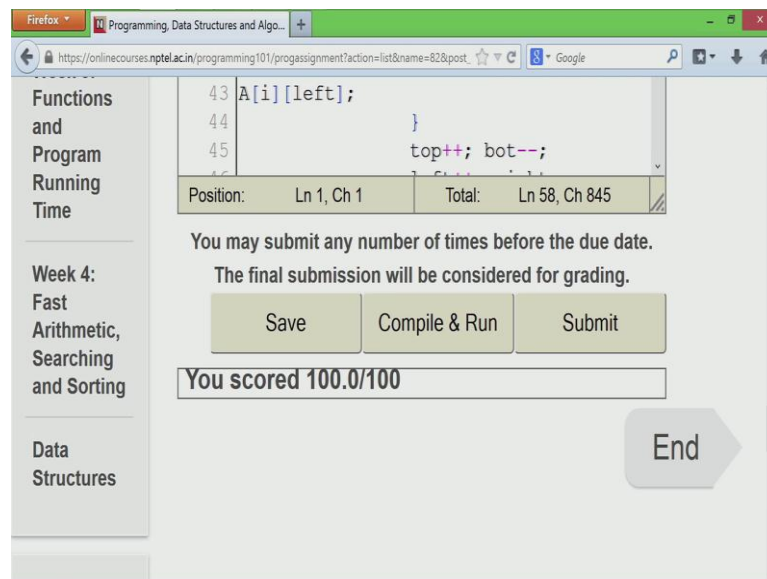
So, all the four test cases are passed, which is a good think to get.

(Refer Slide Time: 24:52)



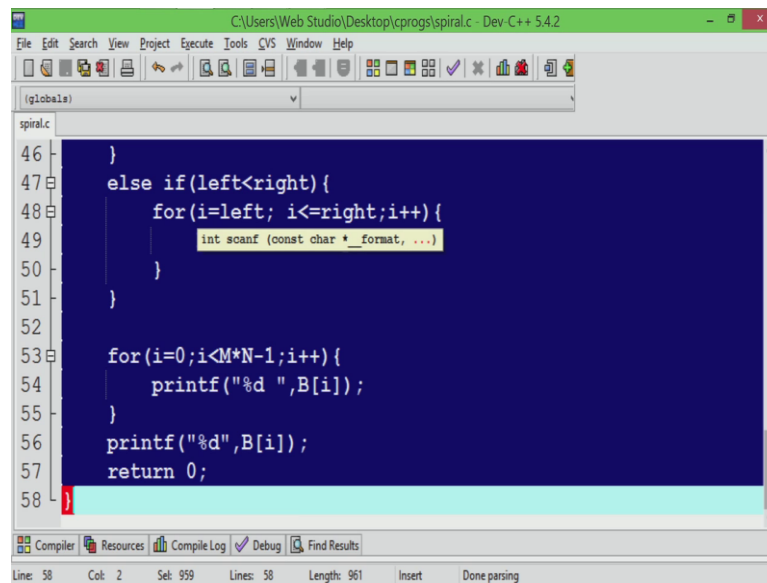
And finally, I do a submit.. So, and I got 100 on 100.

(Refer Slide Time: 25:05)



So, this program seems to be correct.

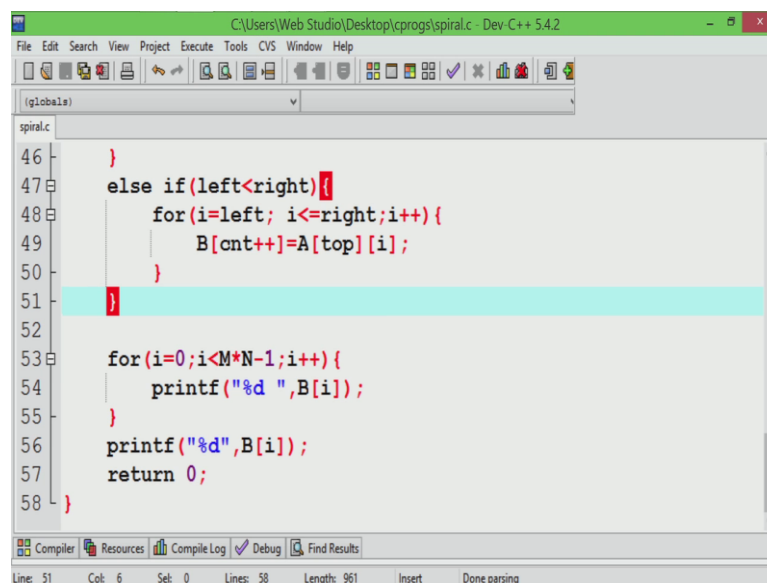
(Refer Slide Time: 25:09)



```
46 }
47 else if(left<right){
48     for(i=left; i<=right;i++){
49         int scanf(const char * __format, ...)
50     }
51 }
52
53 for(i=0;i<M*N-1;i++){
54     printf("%d ",B[i]);
55 }
56 printf("%d",B[i]);
57 return 0;
58 }
```

So, the key thing that I did was not just traversal.

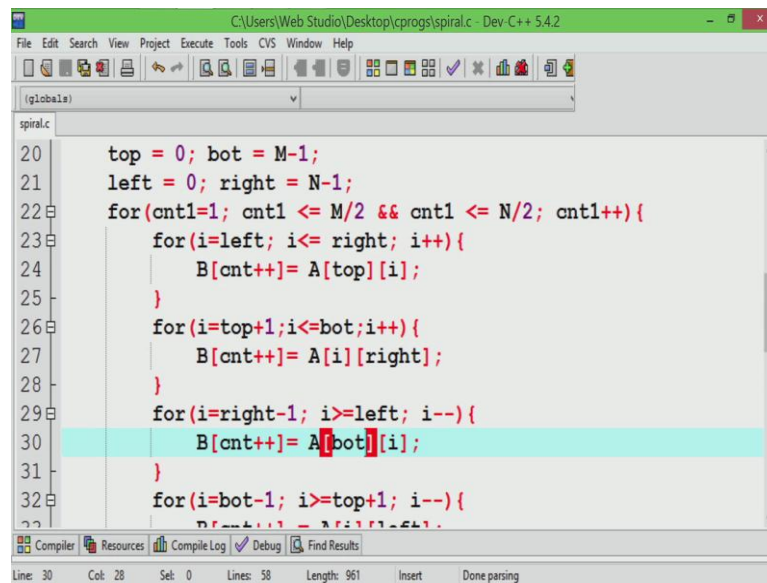
(Refer Slide Time: 25:11)



```
46 }
47 else if(left<right){
48     for(i=left; i<=right;i++){
49         B[cnt++] = A[top][i];
50     }
51 }
52
53 for(i=0;i<M*N-1;i++){
54     printf("%d ",B[i]);
55 }
56 printf("%d",B[i]);
57 return 0;
58 }
```

I want you to look at, what I did with B. So, instead of printing A as similar as I saw the elements, I start it collecting them in B.

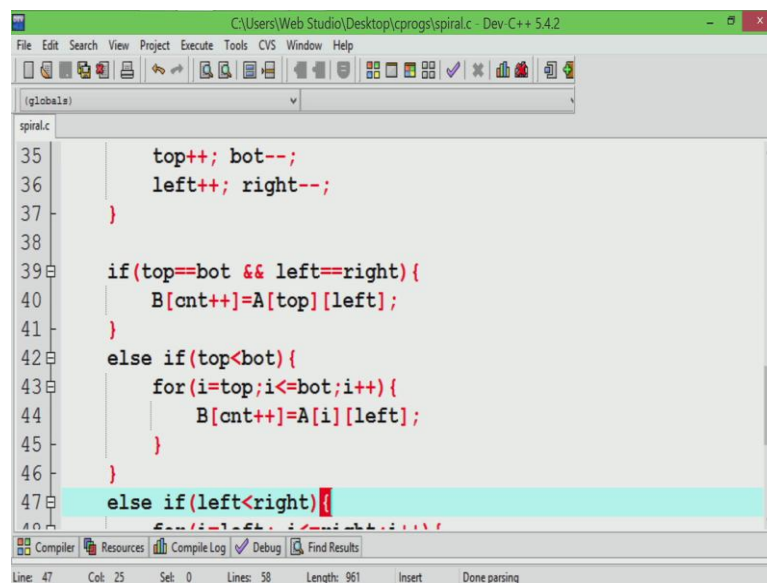
(Refer Slide Time: 25:22)



```
20     top = 0; bot = M-1;
21     left = 0; right = N-1;
22     for(cnt1=1; cnt1 <= M/2 && cnt1 <= N/2; cnt1++){
23         for(i=left; i<= right; i++){
24             B[cnt++] = A[top][i];
25         }
26         for(i=top+1; i<=bot; i++){
27             B[cnt++] = A[i][right];
28         }
29         for(i=right-1; i>=left; i--){
30             B[cnt++] = A[bot][i];
31         }
32         for(i=bot-1; i>=top+1; i--){
33             B[cnt++] = A[i][left];
34         }
35     }
36     top++; bot--;
37     left++; right--;
38 }
39 if(top==bot && left==right){
40     B[cnt++] = A[top][left];
41 }
42 else if(top<bot){
43     for(i=top; i<=bot; i++){
44         B[cnt++] = A[i][left];
45     }
46 }
47 else if(left<right){
```

If you noticed, I start it collecting them in B right from line 24 onwards, I start it collecting them in B.

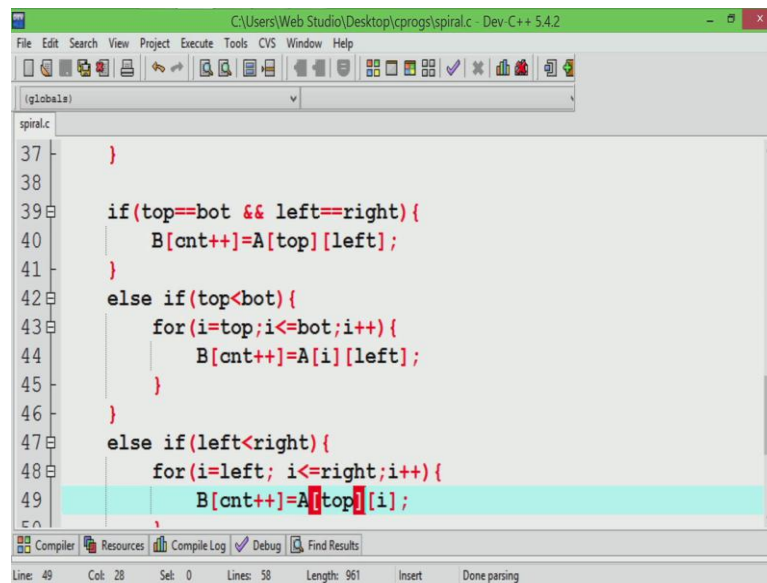
(Refer Slide Time: 25:32)



```
35     top++; bot--;
36     left++; right--;
37 }
38 }
39 if(top==bot && left==right){
40     B[cnt++] = A[top][left];
41 }
42 else if(top<bot){
43     for(i=top; i<=bot; i++){
44         B[cnt++] = A[i][left];
45     }
46 }
47 else if(left<right){
48     for(i=right; i>=left; i--){
49         B[cnt++] = A[bot][i];
50     }
51     for(i=bot-1; i>=top+1; i--){
52         B[cnt++] = A[i][left];
53     }
54 }
```

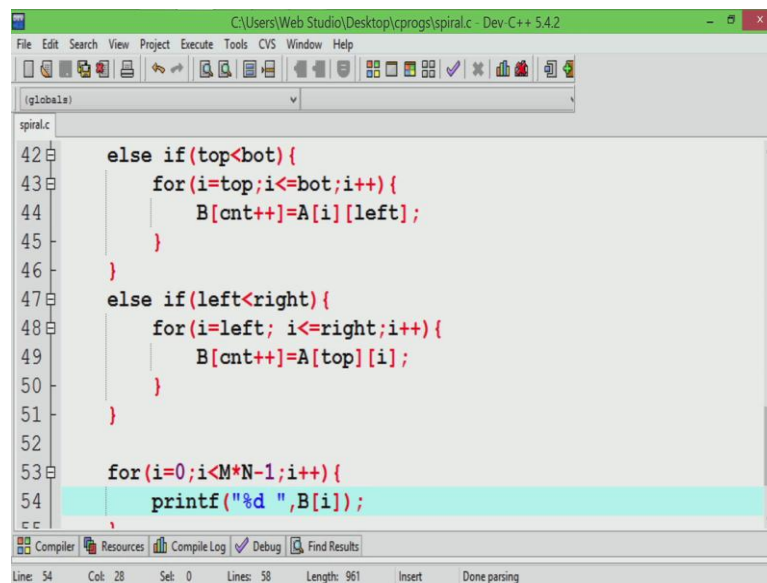
So, at the end of all the traversal my B, the array B would have M times n elements, index from 0 to m into M minus 1.

(Refer Slide Time: 25:35)



```
37     }
38
39     if(top==bot && left==right){
40         B[cnt++]=A[top][left];
41     }
42     else if(top<bot){
43         for(i=top;i<=bot;i++){
44             B[cnt++]=A[i][left];
45         }
46     }
47     else if(left<right){
48         for(i=left; i<=right;i++){
49             B[cnt++]=A[top][i];
```

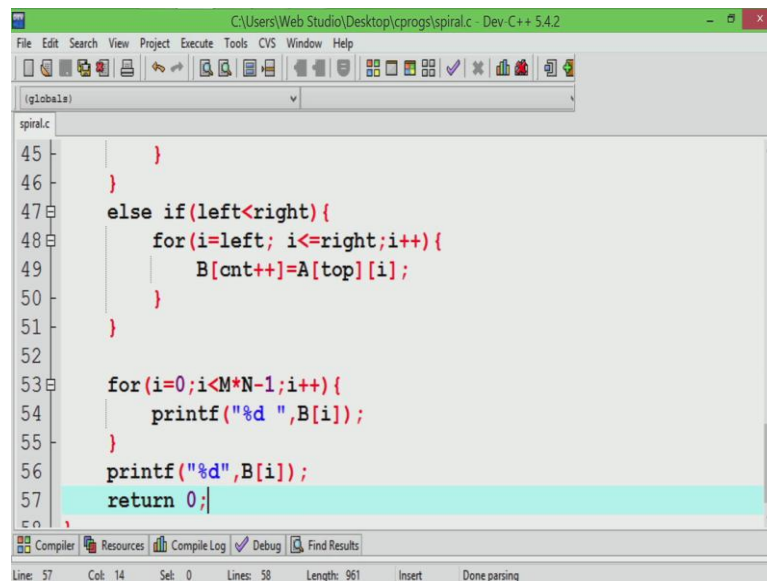
(Refer Slide Time: 25:42)



```
42     else if(top<bot){
43         for(i=top;i<=bot;i++){
44             B[cnt++]=A[i][left];
45         }
46     }
47     else if(left<right){
48         for(i=left; i<=right;i++){
49             B[cnt++]=A[top][i];
50         }
51     }
52
53     for(i=0;i<M*N-1;i++){
54         printf("%d ",B[i]);
```

What I did here is, I printed elements from 0 to M into N minus 2 with a space and printed the M into N minus 1th element without the space, because that is the last one.

(Refer Slide Time: 25:49)



```
45     }
46 }
47 else if(left<right){
48     for(i=left; i<=right;i++){
49         B[cnt++]=A[top][i];
50     }
51 }
52
53 for(i=0;i<M*N-1;i++){
54     printf("%d ",B[i]);
55 }
56 printf("%d",B[i]);
57 return 0;
```

So, I did not do any printing before this. So, this is a nice way to do things, because you are done with earlier work, you just have to concentrate on printing now, you just have to do this one step. So, this is a common way in which you can solve several presentation problems that you been seeing. So, far. Always go and record, what your output must be and take care of the printing, as the last thing in the program. So, this is something that did not touch the logic of the program ever.

So, I believe that many of you had trouble in putting the logic for printing appropriately inside the loop itself. But, I wanted to show that you can do all of this without worrying about the logic. All I did was I saved all of them in an array called B and finally, I printed them outside. So, this is the technique that you can use for many problems in week 3 onwards, I hope you actually adopt this technique from now on.

So, thank you very much and I hope that you enjoy this session as much as I did. So, there are lots of interesting things that happened here, I did some mathematical derivations to show that, you can only run  $M$  by 2 times or  $N$  by 2 times, whichever is smaller and I will leave it you to think, how do I know for sure that either I have only one row left or one column left or exactly one element left. So, go and think about that.

It is a very interesting and challenging problem to think about.

So, I knew that ahead of time, because of the way of doing things. So, maybe you should go and practice yourself on, how to prove something like that and proceed with it.

So, thank you very much and see you next week, bye bye.