Striver DSA A-Z

C++:

① Basic I/O

```
#include <iostream>
Using namespace std;
int main()
{
    a = 5;
    b = 6;
    c = a+b;
  cout << c;
  return 0;
}
```

⇒ It indicate the Program runs successfully.

int namespace std ⇒ It allows you to use Standard library names without the std:: prefix.

```
#include <bits/stdc++.h>
using namespace std;
```

⇒ all c++ library.

#include<bits/stdc++.h>

② Datatypes:

int, float, double, long, string, char

③ If - else:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
  int age;
```

```
cout << " Enter age:";
cin >> age;
if ( age =>18)
{
    cout << " you are Elligble ~~fo~~ to vote :"
}
else
  {
    cout << " Not eligble";
  }
return 0;
```

---

1. If - else
2. If - ~~If else~~ - else if - else
3. nested if

---

4. Switch

~~#include <std. bits/ stdc. h.~~

~~#include <std~~

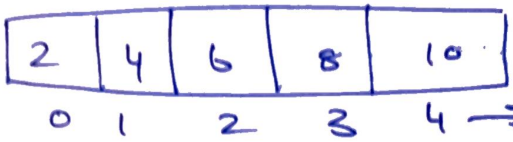
#include <bits/stdc++.h>

Switch:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    cin >> day;
  switch (day)
    {
      case 1:
        cout<< "Monday";
        break;
    case 2:
        cout << "Tuesday";
        break;
    case 3:
        cout << "Wednesday";
        break;
    case 4:
        cout<< "Thursday";
        break;
    case 5:
        cout<< "Friday";
        break;
    case 6:
        cout << "Saturday";
        break;
    case 7:
        cout << "Sunday";
        break;
    default:
        cout << "Invalid";
        break;
    }
  return 0;
}
```

(5.) Array & Strings :

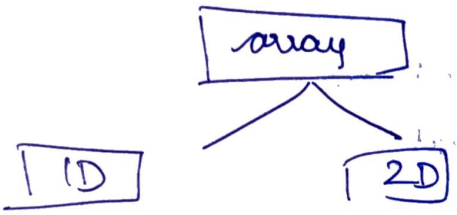
a=[5]



int arr [5] = {2, 4, 6, 8, 10} ⇒ Initialization

Traverse array:

```
for (i=0; i<5; i++)
arr[i];
```



2 Dimensional array:

```
a = [3][5];
```

String:

```
#include <iostream>
using namespace std;
int main()
{
    string s = "Shiva";
    cout<<s[2];
    return 0;
}
```

(6) For loop:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    for (int i=1; i<=10; i++)
    {
        cout << "Striver" << endl;
    }
    return 0;
}
```

i=1 ✓
i<=10 ✓
Striver
i++
i=2 ✓
i<=10 ✓
Striver.

(7) While Loop:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int i=1;
    while (i<=5)
    {
        cout << "Striver" << i << endl;
        i = i+1;
    }
    return 0;
}
```

i=1
i<=5 ✓
Striver 1 ✓
i+1=i
i=2 ✓
2<=5 ✓
Striver 2 ✓
2+1=i
3=i
.
.
.

(8) Do while:

```
#include <bits/stdc++.h>
using namespace std;
int main()
```

```
{
    int i = 2;
    do {
        cout << "Shiva" << i << endl;
        i = i + 1;
       }
    while (i <= 1);
    cout << i << endl;
    return 0;
}
```

9. Functions :

* Functions are set of code which performs something for us.
* Functions are used to modularise code.
* Functions are used to increase readability
* Functions are used to use same code multiple time

Eg: Void → not return anything

return →

Parameterised

non Parameterised.

```
#include <bits/stdc++.h>
using namespace std;
Void printname(string name)
{
  cout << "hey" << name << endl;
}
```

```
int main()
{
    string name;
    cin >> name;
    Printname (name);

    String name2;
    Cin > name2;
    Printname(name2);
    return 0;
}
```

(10.) Pass by reference & Value:

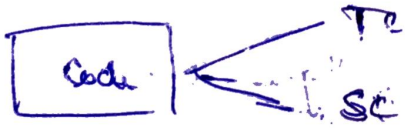
By Value:

```
#include <iostream>
using namespace std;
int square (int);
int main()
{
   int a=5;
   cout << square (a) << endl;
   cout <<" The value of a is" << a << endl;
}
int square (int a)
 {
     return a*a;
 }
```

By reference:

```
#include <iostream>
using namespace std;
int
Void squarebyref(int &); // declaration
int main()
{
   int b = 4;
   squarebyref(b);
   cout<<" The value of b is"<<b;
}
Void (s
Void squarebyref(int &b) //defination
{
    b = b*b;
}
```

Time complexity:



Big oh notation ⟶ ⟶ Time taken

Time complexity:

Big oh notation ⇒ O( )
↳ Time Take

Rules:

* TC, worst case sinario
* avoid constants
* avoid lower ~~Case~~ values

Eg: 1

```
for (i=1 ; i<=5 ; i++)
{
    cout << "Raj";
}
```

O(5 x 3)

O(5)

O(n x 3)

Eg 2:

```
If (mark < 25) cout << "grade-D";
else if (mark <45) cout << "grade c";
else if (mark <65) Cout << "grade 'B';
else               cout << "grade A";
```

Best case	Average	Worst Case
Eg Mark = 10		Eg: Mark = 70
O(2)		O(4)

Avoid constants

Eg: $O(4N^3 + 3N^2 + 8)$ $N = 10^5$

$\Rightarrow O(4 \times 10^{15} + 3 \times 10^{10} + 8)$ → avoid (constan

→ avoid lower valu

Big Oh (O)	Theta Θ	Omega (
* worst case		
* upper bound	Average complexity	lower bo

Question:

```
for (int i=0; i<N; i++)
{
    for (int j=0; j<N; j++)
    {
        // code block
    }
}
```

1 2 6 3 4 5

$O(n \times$

Time complexity:

i=0 {j=0, 1, 2, 3 ... N}
i=2 {j=0, 1, 2, 3 ... N}
i=3
⋮
i=n {j=0, 1, 2, 3, ... N}

N+1

n x n

$\Rightarrow n^2$

$\Rightarrow O(n^2)$

Eg 2:

```
for(i=0; i<N; i++)
{
    for(j=0; j<=i; j++)
    {
        //code
    }
}
```

Time complexity:

$i=0$ {$j=0$}

$i=1$ {$j=0,1$}

$i=2$ {$j=0,1,2$}

$i=n-1$ {$j=0,1,2 \ldots n-1$}

$(1+2+3+4 \ldots n)$

$$\frac{n \times (n+1)}{2} \Rightarrow \frac{n^2}{2} + \frac{n}{2}$$

$$\Rightarrow O\left(\frac{n^2}{2}\right) = O(n^2)$$

Space complexity: ⇒ memory space

Auxiliary space + Input space

↳ The space that you need to solve the problem

↳ The space that you need to take to store the Input.

(a) --- (b) → Input space

(c) = a + b

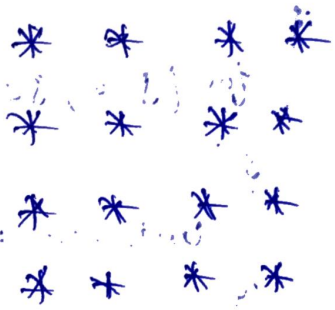
↳ Auxiliary space

Space complexity O(n) O(3)

~~O(n)~~ O(10^8) → Time complexity.

Patterns: (22) Questions:

* Nested loops



Outer loops (rows) — Inner loops (columns)

1. for the outer loop, count the number of lines.
2. for the Inner loop, focus on the columns, and connect them somehow to the rows.

(3.) Print the '*' inside the inner for loop.

(4.) Observe symmetry [optional]

Eg:


```
for (i=0; i<4; i++)
{
    for (j=0; j<4; j++)
    {
        cout << "*";
    }
    cout << endl;
}
```

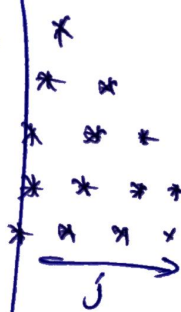
In an online compiler:- The code need to run according to their test cases.

```
#include <bits/stdc++.h>
using namespace std;
~~void Pair~~
Void Pattern1(int n)
{
```

```
for ( i=0; i<=n; i++) // rows
    {
      for (j=0; j<=n; j++) // column
        {
          cout << "*";
        }
      cout << endl;
    }
}
int main()
 {
    int t;
    cin >> t;
    for ( int i = 0; i<t ; i++)
    {
       int n;
       cin >> n;
       Pattern1( n);
    }
}
```

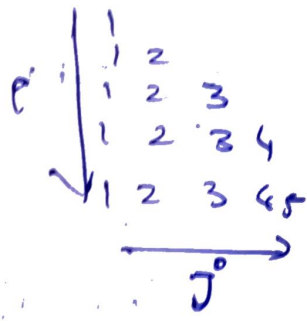
Pattern 2:

```
# include <bits/stdc++.h>
using namespace std;
void pattern2( int n)
{
    for (i=0; i<n ; i++) // rows
        {
            for (j=0; j<i ; j++) // column
```



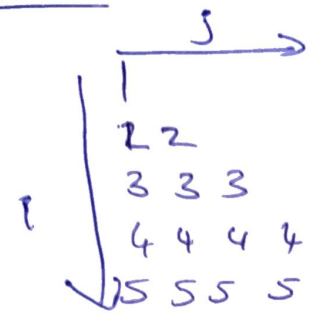
```
    {
        cout << "*" ;
    }
    cout << "endl";
  }
}
int main()
{
    int n;
    cin >> n;
    Pattern(n);
}
```

Pattern 3:




```
#include<bits/stdc++.h>
using namespace std;
int i, j;
int main()
{
    for(i=1; i<=5; i++) //rows
    {
        for(j=1; j<=i; j++) //column
        {
            cout << j << " ";
        }
        cout << endl;
    }
}
```

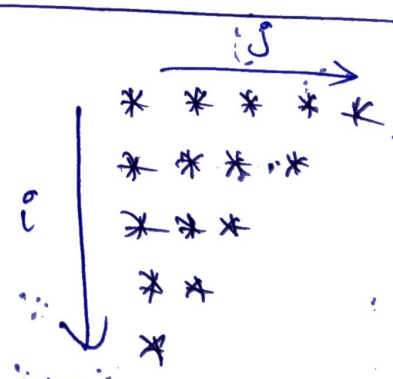
Pattern 4:




```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    for (i=0; i<=5; i++) //rows
    {
        for(j=1; j<=i; j++) //column
        {
            cout << i << " ";
        }
        cout << endl;
    }
}
```

Pattern 5: n - row + 1

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    for (i=0; i<=n; i++) //row
    {
        for (j=0; j<=n-i+1; j++) //column
        {
```

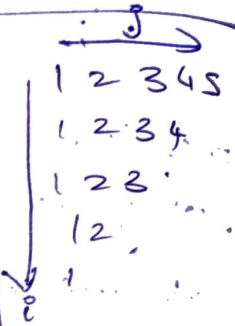


Observation:

1 → 5	(n - row + 1)
2 → 4	5 - 1 + 1 = 5
3 → 3	5 - 2 + 1 = 4
4 → 2	5 - 3 + 1 = 3
5 → 1	

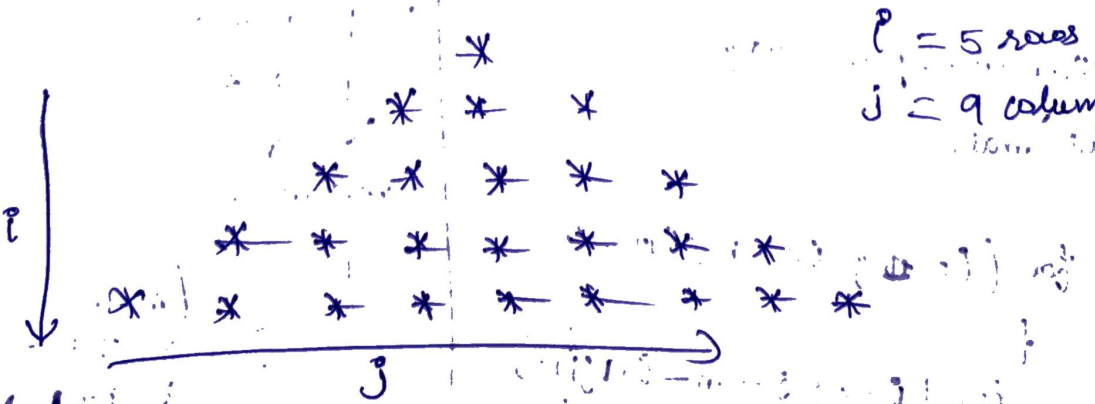
```
        cout << "*";
    }
    cout << endl;
  }
}
```

Pattern 6:




```
#include <bits/stdc++.h>
using namespace std;
int main()
{
  for (i=1 ; i<=5; i++) //rows
  {
    for (j=1; j<=n-i+1; j++)//columns
    {
      cout << j << " ";
    }
    cout << endl;
  }
}
```

Pattern 7:



Space , Star , Space

row 1 [4 , 1 , 4]

row 2 [3 , 2 , 3]

row 3 [2 , 5 , 2]

row 4 [1 , 7 , 1]

row 5 [0 , 9 , 0]

$n - i - 1$ $n - i - 1$

$5 - 0 - 1 = 4$

$5 - 1 - 1 = 3$

$5 - 2 - 1 = 2$

$5 - 3 - 1 = 1$

$5 - 4 - 1 = 0$

$2 \times i + 1$

$2 \times 0 + 1 \Rightarrow 1$

$2 \times 1 + 1 \Rightarrow 3$

$2 \times 2 + 1 \Rightarrow 5$

$2 \times 3 + 1 \Rightarrow 7$

$2 \times 4 + 1 \Rightarrow 9$

~~#include <bits/stdc++.h>~~

#include <bits/stdc++.h>

using namespace std;

~~int main() int n;~~

{

void Pattern7()

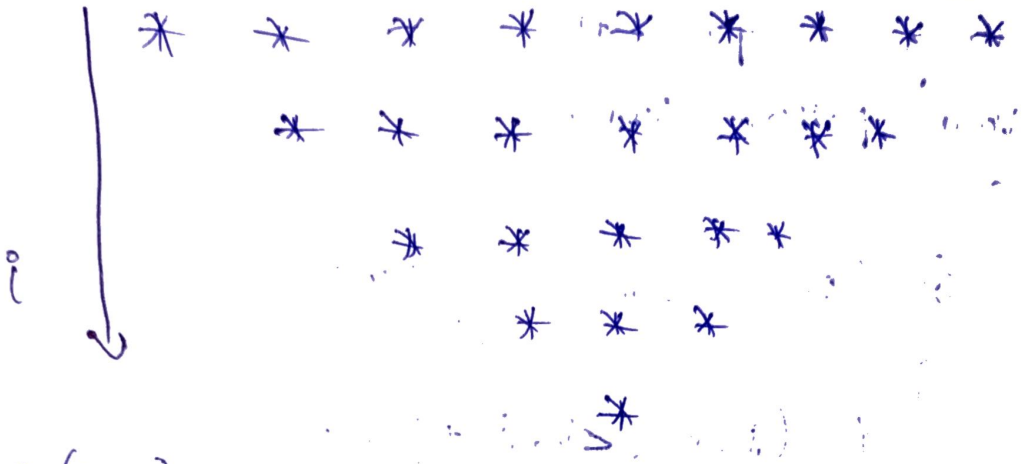
{

for (int i=0; i<=n; i++)

{

```
#include <bits/stdc++.h>
using namespace std;
void pattern7(int n)
{
    for (int i=0; i<=n; i++)  //rows
    {
        for (int j=0; j<2*n-i-1; j++) //space
        {
            cout<<" ";
        }
        for(int j=0; j<2*i+1; j++) //star
        {
            cout<<"*";
        }
        for (int j=0; j<n-i-1; j++) //space
        {
            cout<<" ";
        }
    }
}
int main()
{
    int n;
    cin>>n;
    pattern7(n);
}
```

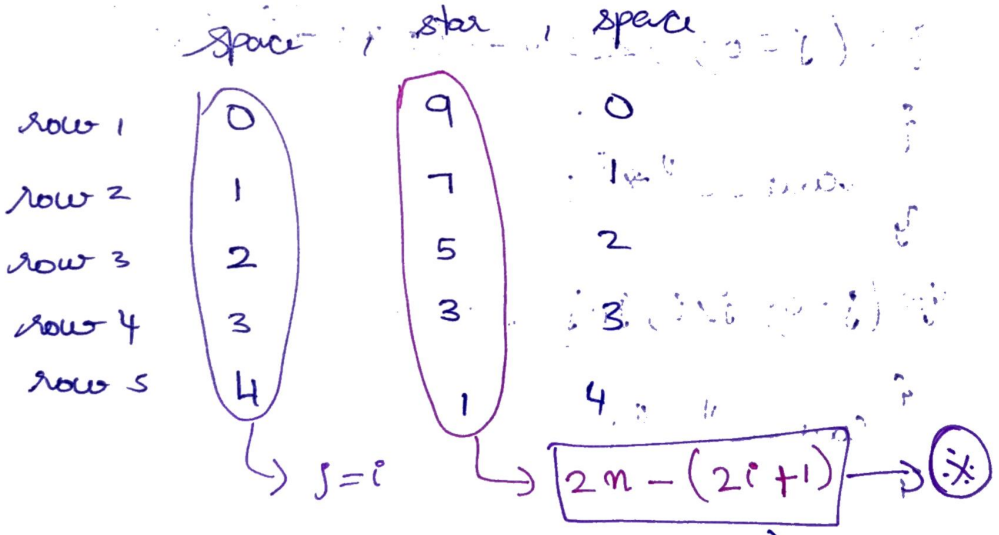
Pattern 5:



i (row) = 5

j (column) = 9

Pattern:



(1.) $2(5)-(2\times0+1)$

$\Rightarrow 10-1=9$

(2.) $2(5)-(2\times1+1)$

$=10-3=7$

(3.) $2(5)-(2\times2+1)$

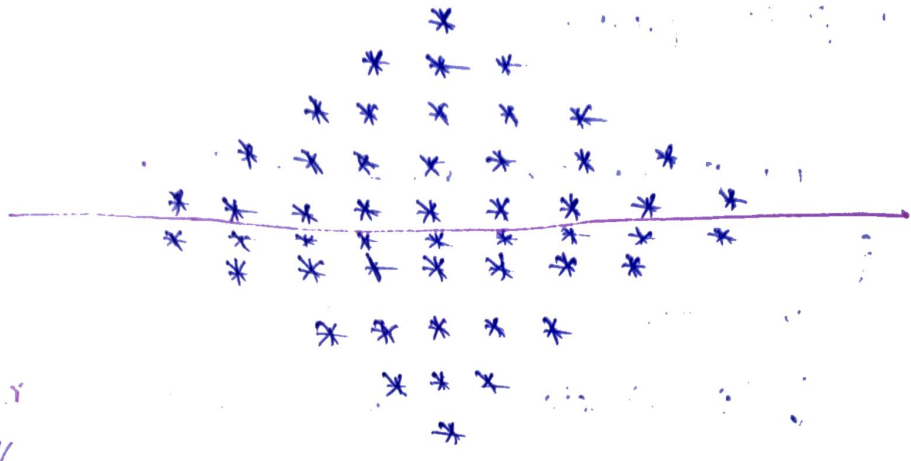
$=10-5=5$

(4.) $10-7=3$

(5.) $10-9=1$

```
#include <bits/stdc++.h>
using namespace std;
void patterns(int n)
{
  for (i=0; i<n; i++) //rows
  {
    for (j=0; j<i; j++) //space
    {
      cout<<" ";
    }
    for (j=0; j<2n-(2*i+1); j++) //star
    {
      cout<<"*";
    }
    for (j=0; j<i; j++) //space
    {
      cout<<" ";
    }
    cout<<endl;
  }
}
int main()
{
  int n;
  cin>>n;
  patterns(n);
}
```

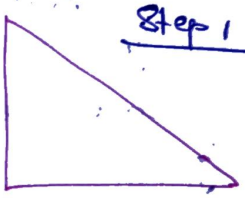
Pattern 9:



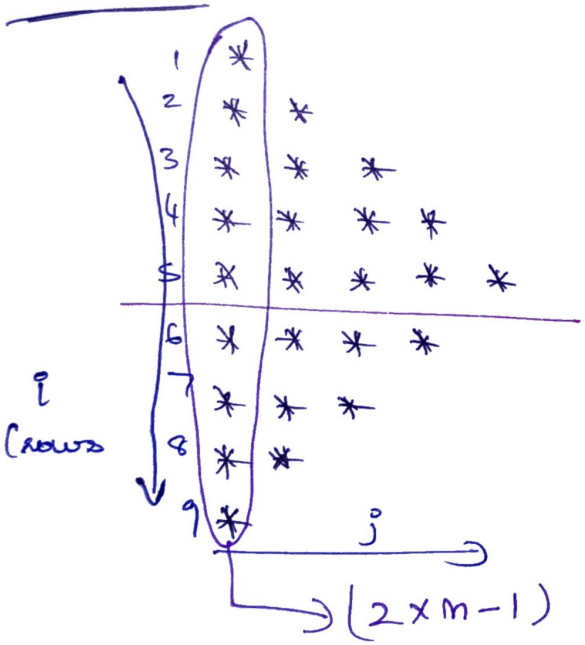
// combine Pattern 7 & Pattern 8.

Pattern 10:

Observe Pattern



Step 1:



i = 9 rows

j = 5 columns

Step 2:

(We need to flip the triangle)

* Stop on the 5th and ~~tab~~ break the loop
* $\boxed{2n-i}$ → formula

```
# include <iostream>
using namespace std;
void pattern1(int n)
{
  for (int i = 1; i <= 2*n-1; i++) // no of rows
    {
      int stars = i; // declaration  star = i
      if (i > n) stars = 2*n-1; // break after n rows
      for (j=1; j<=stars; j++)
      {
        cout << "*";
      }
      cout << endl;
    }
}
int main()
{
    int n;
    cin >> n;
    pattern1(n);
}
```

```
# include <iostream>
using namespace std;
void ~~main()~~ Pattern1(int n)
{
  // upper part of the triangle
  for (int i = 1; i <= n; i++) // rows
  {
```

Chatgpt

```
for (int j = 1; j <= i; j++) column
  {
    cout << " *";
  }
  cout << endl;
}
    // lower Part of the triangle
for (int i = n-1; i >= 1; --i) // rows
  {
    for (int j = 1; j <= i; ++j // column
    {
      cout << " *";
    }
    cout << endl;
  }
}
int main()
  {
    int n;
    cin >> n;
    Pattern10(n);
  }
```

Pattern 11.

```
1 1
2 0 1
3 1 0 1
4 0 1 0 1
5 1 0 1 0 1
```

Observe Pattern:

Even rows start with 0

for (int i = 0; i < n; i++)

if (i % 2 == 0) start = 1; } rows

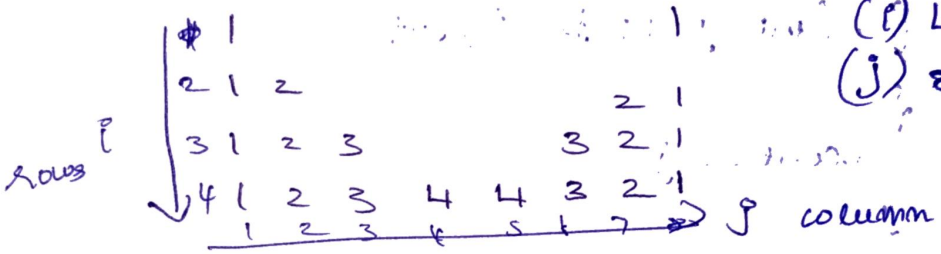
else start = 0;

for (int j = 0; j < i; j++)

```
#include <iostream>
using namespace std;
void Pattern11(int n)
{
  int start = 1; // start with number 1
  for (int i = 0; i < n; i++) // no of rows
  {
    if (i%2==0) start = 1; // 1st row start
    else start=0;          with 1 or 0
    for (int j = 0; j <= i ; j++) // column
    {
      cout << start;
      start = 1 - start; // flip 1 into 0
    }
  cout << endl;
  }
}
```

```
int main()
{
  int n;
  cin>>n;
  pattern11(n);
}
```

Pattern12:



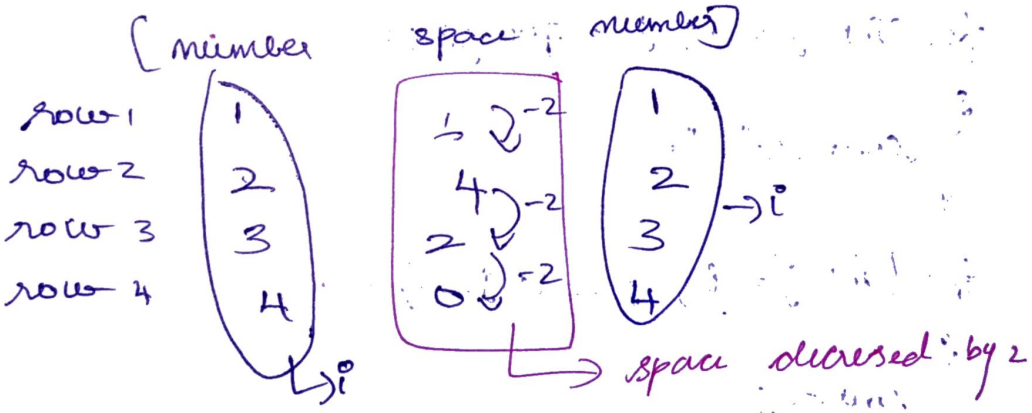


figure out formula for space:

1st n = 2x(n-1)
= 2x(4-1)
= 2x3 = 6

2nd num ⇒ 2x(n-1)
⇒ 2x(3-1)
⇒ 4

3rd n ⇒ 2x(2-1)
= 2

4th ⇒ 2x(1-1)
= 0

```
#include <bits/stdc++.h>
using namespace std;
void pattern12(int n)
{
    int space = 2*(n-1);
    for (int i=1; i<=n; i++) //rows
    {
        for (int j=1; j<=i; j++) //space
        {
            cout<<j;
        }
        for (int j = i; j<=space; j++) //numbers
        {
            cout <<" ";
        }
        for (int j = i; j>=1; j--) #space
        {
            cout<<j;
        }
        cout << endl;
        space -= 2; // Decrease space by 2 for the next line
    }
};
int main()
{
    int n;
    cin >>n;
    pattern12(n);
}
```

```
for j=i; j; j>=1; j--
j=i
j>=1; j--
```

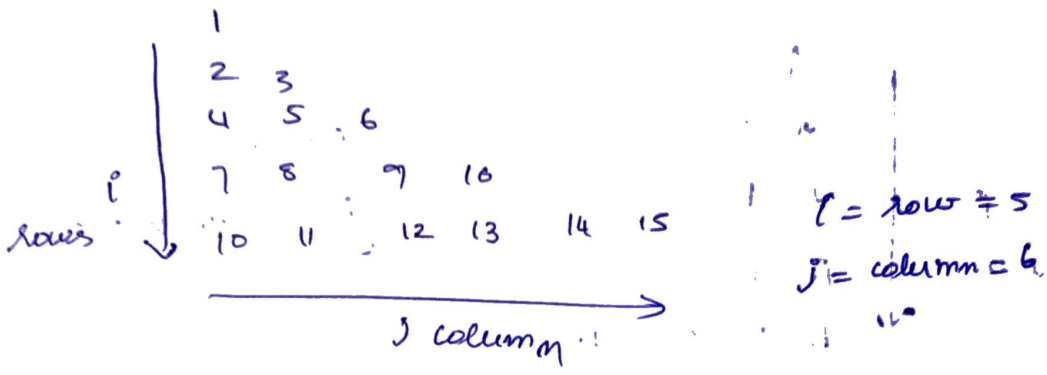
1

2 1

2>=1; j--

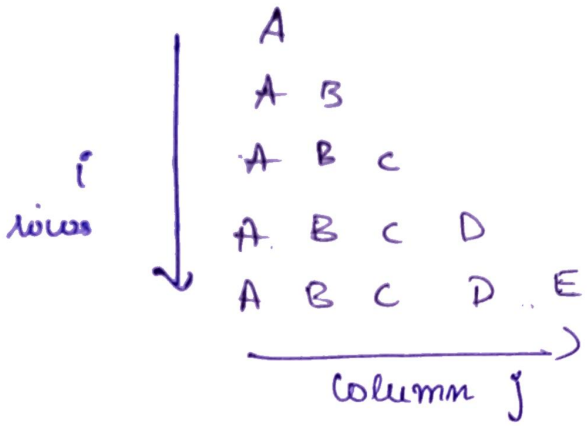
2

Pattern 13 :




```
#include <iostream>
using namespace std;
void Pattern13 (int n)
{   int num = 1; // declaration
  for(int i=1; i<=n; i++) // rows
  {
    for( int j=1; j<=i; j++) // column
    {
      cout<< num <<" "; // Printing output
      num = num +1; // Increment
    }
  cout << endl;
}
int main()
{
    int n;
    cin >>n;
    Pattern13(n);
}
```

Pattern 14:

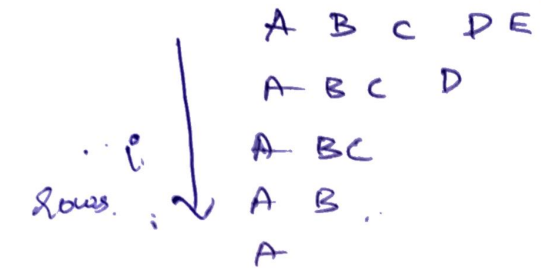


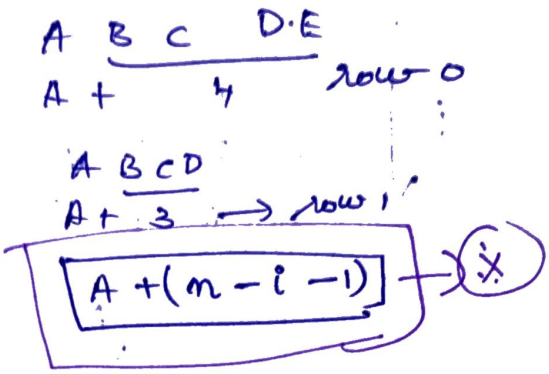
i = row = 5

j = column = 5

```
#include <bits/stdc++.h>
using namespace std;
Void Pattern14 (int n)
{
  for (int i = 0; i < n; i++) // row
  {
    for (char ch = 'A'; ch <= 'A' + i; ch++) // column
    {
      cout << ch << " ";
    }
    cout << endl;
  }
int main()
  {
    int n;
    cin >> n;
    Pattern14(n);
  }
```

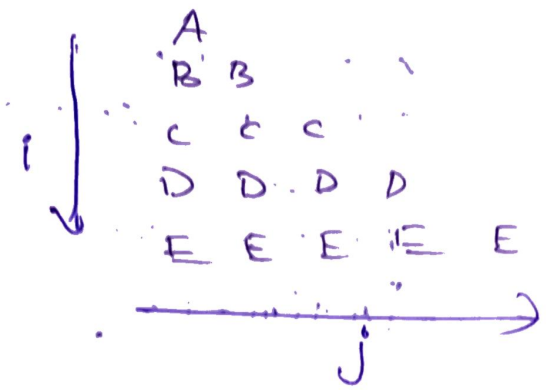
Pattern is :






```
#include <iostream>
using namespace std;
void patternIs(int n)
{
  for(i=0; i<n; i++) //rows
  {
    for (char ch = 'A'; ch<='A' + (n-i-1); ch++)
    {
      cout << ch << " ";
    }
    cout << endl;
  }
}
int main()
{
  int n;
  cin >> n;
  PatternIs(n);
}
```

Pattern 16:



0 → A A+i
0 → B A+i

```
#include <iostream>
using namespace std;
void pattern16(int n)
{
  for (i=0; i<n; i++) // rows
  {
    char ch = 'A' + i; // declaration / Formula
    for (j=0; j<=i; j++) // J → columns
    {
      cout << ch << " ";
    }
    cout << endl;
  }
}
int main()
{
  int n;
  cin >> n;
  pattern16(n); }
```

Pattern 17:

```
        A
      A B A
    A B C B A
```

i = row = 5

j = column = 9

Rows ↓ i

Columns → j

	Space	Char	space
row 1	3	1	3
row 2	2	3	2
row 3	1	5	1
row 4	0	7	0

Space → n-i-1

space → n-i-1

Char → 2×i+1

5 / 2 +1 ⇒ 3

A B C B A

++ ++ -- --

```
#include <iostream>
using namespace std;
void pattern16(int n)
{
  for (int i=0; i<n; i++) // rows
  {
    for (int j = 0; j < 2*i-1; j++) // space
    {
      cout << " ";
    }
  char ch = 'A'; // char
  int breakpoint =(2*i+1)/2; // To print reverse declaration
```

```
for ( int j = 1 ; j <= 2 * i + 1 ; j++)   // Print ch
{
    cout << ch;
    if (j <= breakpoint) ch++;
    else ch--;
}                                          // condition & Print reverse

for (int j=0; j < n-i+1; j++)              // space
{
    cout << " ";
}

cout << endl;
}
}
int main()
{
    int n;
    cin >> n;
    Pattern8(n);
}
```

Pattern 18:

```
E
D E
C D E
B C D E
A B C D E
```

i = row = 5

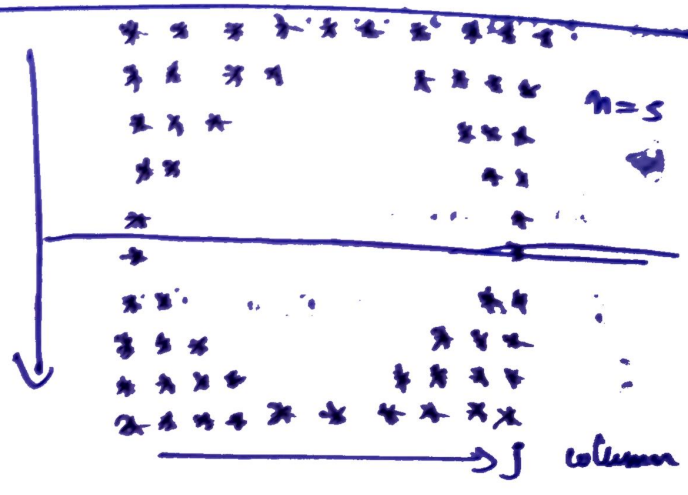
j = column = 5

i

j

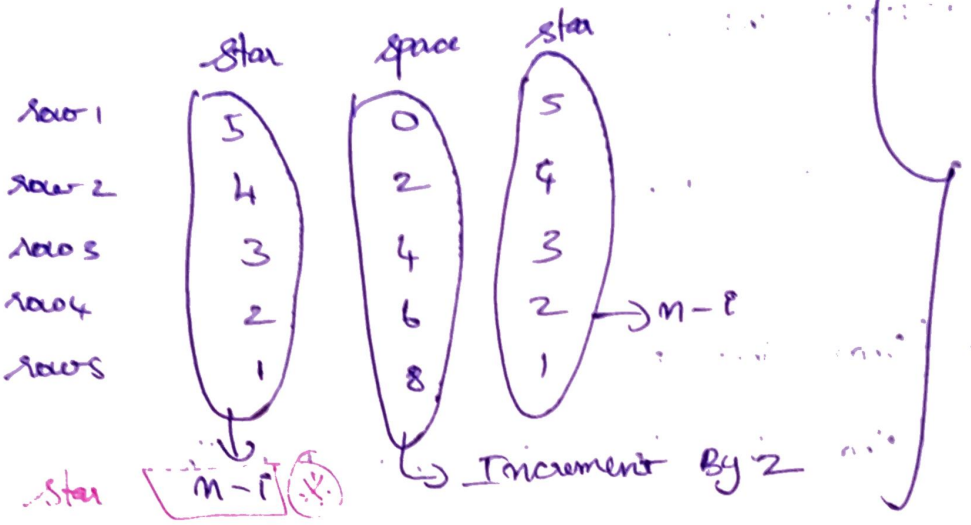
```
#include <bits/stdc++.h>
using namespace std;
void Pattern18(int n)
{
    for(i=0; i<n; i++) //rows
    {
        for(char ch='E'-i; ch<='E'; ch++) //column
        {
            cout<<ch<<" ";
        }
        cout<<endl;
    }
}
int main()
{
    int n;
    cin>>n;
    Pattern18(n);
}
```

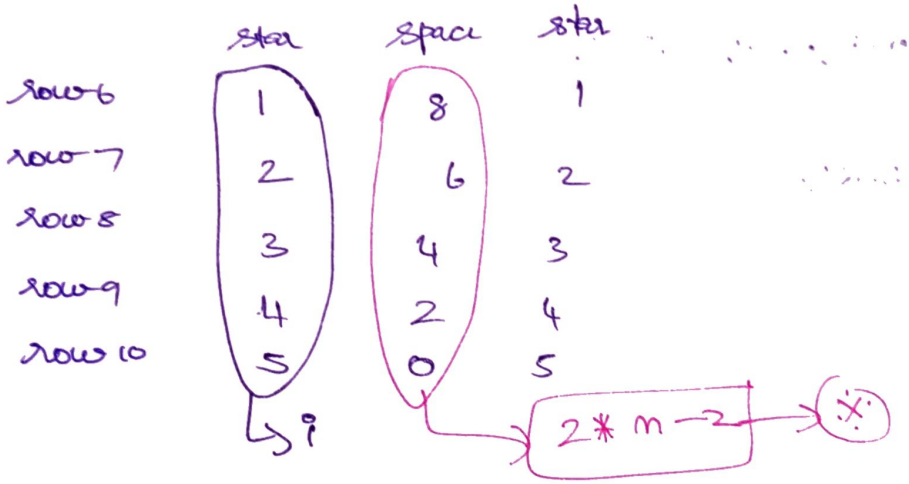
Pattern 19:



row = i = 10

column = j = 10



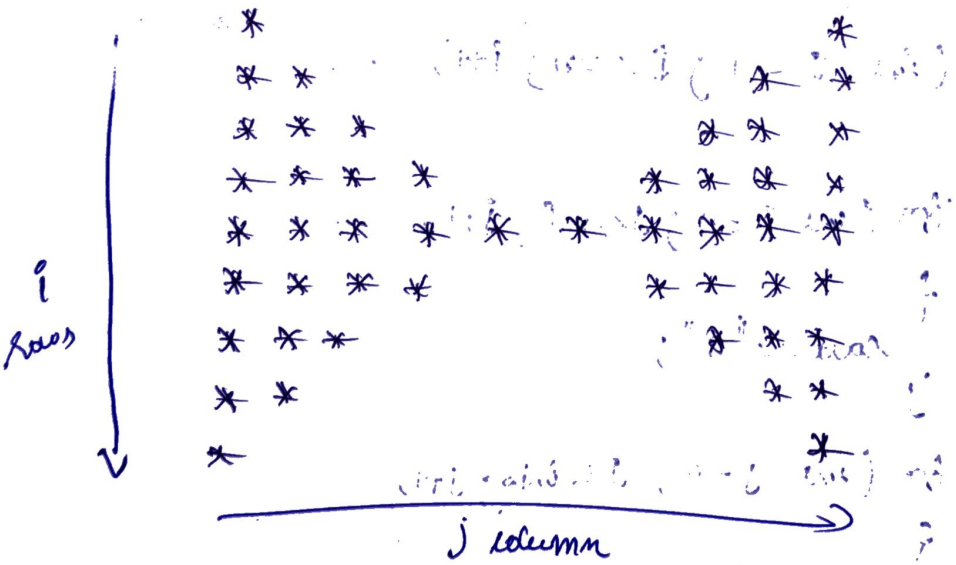



```
#include <iostream>
using namespace std;
void Pattern19(int n)
{
    int cnts = 0; // initial space(cnts) = 0
    for (int i = 0; i < n; i++) //row
    {
```

```
for (int j=1; j<=n-i; j++) // star
  {
    cout << " * ";
  }

for (int j=0; j<inis; j++) //space
  {
    cout << " ";
  }

for (int j=1; j<=n-i; j++) //star
  {
    cout << " * ";
  }

inis += 2; // space increment by 2

cout << endl;
}

inis = 8; // initial space (inis) = 8 (or) use formula 2*n-2

for (int i=1; i<=n; i++) //rows
{
  for (int j=1; j<=i; j++) //star
  {
    cout << " * ";
  }

  for (int j=0; j<inis; j++) //space
  {
    cout << " ";
  }
```

```
for (int j=1 ; j<=i ; j++)   // star
  {
    cout << " *";
  }
  ins -= 2;   // decrement by 2 spaces.
  cout << endl;
 }
}
int main()
 {
   int n;
   cin >> n;
   Pattern19(n);
 }
```

Pattern 20:



i = 9
j = 10

for Example n=5 or n=10

2n-1 times this will loop

	star	space	star
	1	8	1
	2	6	2
	3	4	3
	4	2	4
	5	0	5
	4	2	4
	3	4	3
	2	6	2
	1	8	1

if (row == n/2)
space += 2
star = 2n-1

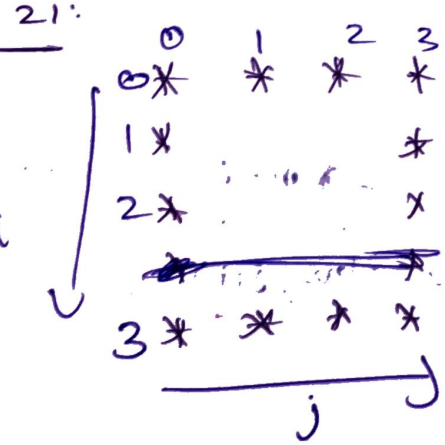
```
#include <iostream>
using namespace std;
void pattern20(int n)
{
    int spaces = 2*n-2; // declaration for space
    for (int i =1; i<=2*n-1; i++) //rows
    {
    int stars = i;
    if (i>n) stars = 2*n-i; // condition for Printing star after n
    for (int j =1; j<=stars; j++) //star
    {
      cout << "*";
    }
```

```
for (int j=1 ; j <= spaces ; j++) //space
{
    cout << " ";
}

for (int j=1 ; j <= star ; j++) //star
{
    cout << "*";
}

cout << endl;
if (i < n) spaces += 2;
else space += 2;
}                    // condition for space increment & Decrement
}
int main()
{
    int n;
    cin >> n;
    Pattern20(n);
}
```

i < n spaces += 2
i < 5 space increment
6 < 5 X

i < n space -= 2
i < 5 space decrement
6 < 5 X
↓
else
space Increment.

Pattern 21:



i = row = 4
j = row = 4
[square]

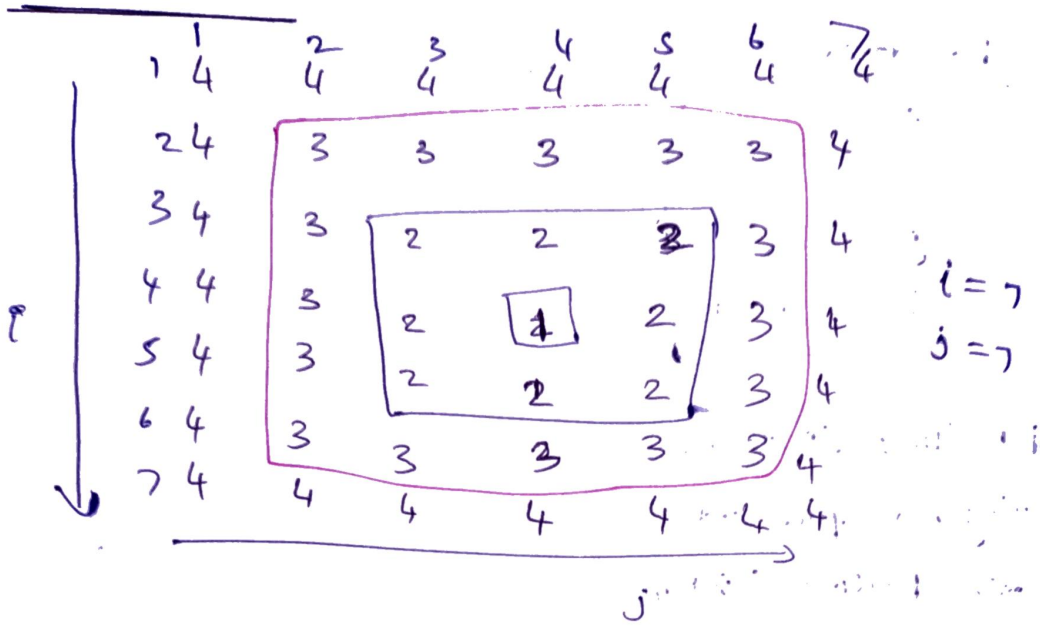
Try to fill stars in boundaries

boundaries:

```
i = 0
i = n-1
j = = 0
j = = n-1
```

```
#include <iostream>
using namespace std;
void Pattern21 ( int n)
{
   for (int i =0; i<n; i++) // rows
   {
      for ( int j =0 ; j<n; j++) // column
      {
         // boundaries condition to Print star
         if ( i==0 || j ==0 || i == n-1 || j== n-1)
      {
         cout << "*";
      }
      else cout<< " ";
   }
   cout << endl;
}
}
int main()
 {
   int n;
   Cin >>n;
   Pattern21 (n); }
```

Pattern 22:



n – value ⇒ new matrix 1

n – new matrix 1 ⇒ new matrix 2

n – new matrix 2 ⇒ new matrix 3

n – new matrix 3 ⇒ new matrix 4

$2 \times n - 1$ → rows formula / column formula

~~2 * n - 1~~

```
#include <bits/stdc++.h>
using namespace std;
void pattern22 (int n)
{
    for (int i=0; i<2*n-1; i++) // rows
    {
        for (int j=0; j<2*n-1; j++) // columns
```

// Initialising the top, down, left, right indices of a cell.

```
{
  int top = 1;
  int bottom = 5;
  int right = (2 * n - 2) - 5;
  int left = (2 * n - 2) - 1;
```

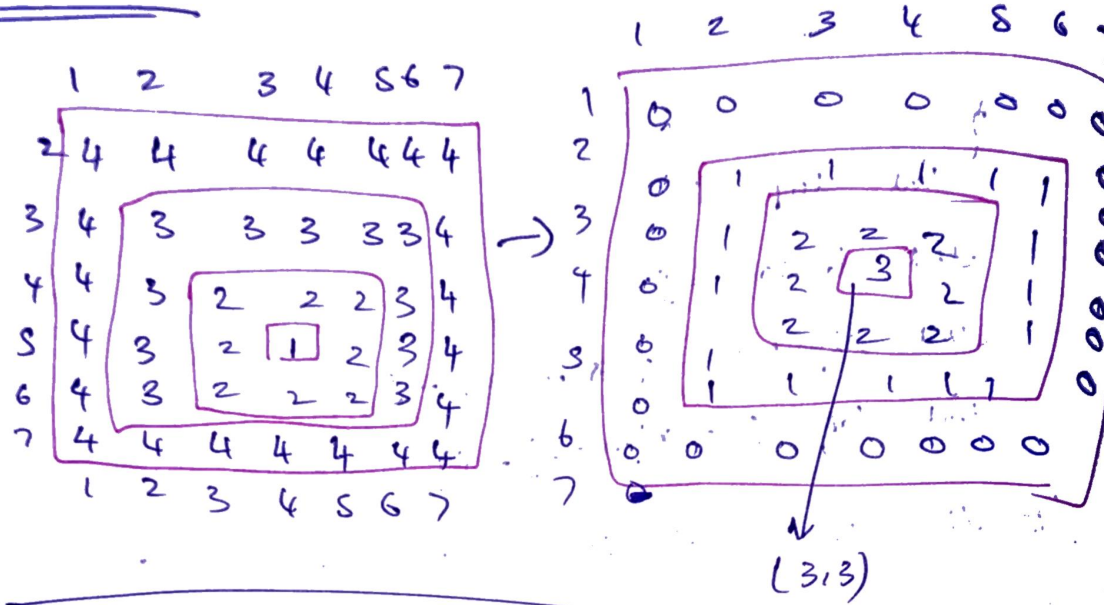
/* Min of 4 directions and then we subtract from n because Previously we would get a pattern whose border has 0s, but we want the border Ns and then decreasing inside; */

```
  cout << (n - min(min(top, bottom),
      min(left, right))) << " ";
}
```

/* As soon as the numbers for each iteration are printed, we move to the next row and give a line break otherwise all numbers would get printed in 1 line. */

```
cout << endl;
}
int main()
{
  int n;
  cin >>
  Pattern22(n);
}
```

Explanation:



top = i

left = j

right = (2n − 1) − 1 − j

bottom = (2n − 1) − 1 − i