10) Pattern -10 Fancy Pattern # 3

```
*
* 1 *
* 1 2 1 *
* 1 2 3 2 1 *
* 1 2 1 *
* 1 *
*
```

} Pattern-1

} Pattern-2

The digits are like palindrome. First let's see how can we print the below pattern.

```
i=0   1                                   (1)
i=1   1   2   1                           (3)
i=2   1   2   3   2   1                   (5)
i=3   1   2   1                           (3)
i=4   1                                   (1)
```

} odd number

j < 2*i+1

j < 2*i+1 for the growing phase only

* After printing pattern-1 and 2, we have to add one condition for 1st & last column to add a star.

Pattern-1

```
for (int j=0 ; j<2*i+1 ; j++) {
        if (i<=(n/2)) {
            if (j<=i)
                cout << j+1;
            else cout << 2*i-j+1;
        }
}
```

Pattern-2

```
for (int j=0 ; j<=2*(n-i-1) ; j++)
        if (i>(n/2)) {
            if (j<=i) cout << j+1;
            else cout << 2*i-j+1;
        }
```

11) Pattern-11 Fancy Pattern #2

```
1
2 * 3
4 * 5 * 6
7 * 8 * 9 * 10
7 * 8 * 9 * 10
4 * 5 * 6
2 * 3
1
```

Pattern-1

Pattern-2

Pattern-1

We will declare a variable num & initialize it with 1 as we have to print the counting. Also we can observe that each row has odd number of characters & hence inner for loop will run for odd number of times. Also when the col-number is even then only

number is printed else we print *.

Code for pattern-1 int num = 1;

```
for (int i=0; i<n; i++) {
    for (int j=0; j<2*i+1; j++) {

        if(j%2 == 0) {
            cout << num << " ";
            num++;
        }
        else {
            cout << "* ";
        }
    }
    cout << endl;
}
```

Pattern-2

```
           0 1 2 3 4 5 6
           7 * 8 * 9 * 10   i=0
10-6 ←     4 * 5 * 6        i=1
 =4        2 * 3            i=2
6-4=2      1                i=3
```

i=0	j = 6	2 x 4 − 1 = 7
i=1	j = 4	2 x 4 − 2x1 − 1 = 5
i=2	j = 2	2x4 − 2x2 − 1 = 3
i=3	j=0	2x4 − 2x3 − 1 = 1

j < 2*n − 2*i − 1; Condition found.

If j is even then only print number else print star.

10−6 = 4 } k = k−j
6−4 = 2

Code

```
int k = 2* n - 1 ;
for (int i=0; i<n; i++) {
        int j=0;
        for (          ; j<2*n-2*i-1;j++){
                if(j%2==0){
                        cout << k << " ";
                        k++; }
                else { cout << "*  "; } }
        k=k-j; cout << endl; }
```

```
        num++;
    }
```

13) Pattern-13 Pascal's Triangle Pattern.

```
1
1  1
1  2  1
1  3  3  1
1  4  6  4  1
1  5  10 10 5  1
1  6  15 20 15 6  1
```

```
      1
    1   1
  1   2   1
1   2   1
```
Fig 1

Observation from this is that the below number is made up of the sum of the 2 adjacents in the above row.
See in Fig 1 in row 3, 2 is made from 1+1 = 2.
3 → 1+2 = 3
3 → 1+2 = 3

```
      1
    1 2 1
  1 3 3 1
```

Now there is a formulae of binomial co-efficient to print the digits in the pascal triangle.
The formulae is `C = C*(i-j)/j;` where i is varying from [1, N] and j is varying from [1, i].

We won't be going into the derivation of this formulae as it will be a little complex.

Code

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

Dry run on below example

```
1
1   1
1   2   1
```

1st iteration (i = 1)

j = 1 1 <= 1 (True) 1

Print 1

j = 2, 2 <= 1 (False)

2nd iteration (i = 2)

```
1
1 1
```

j = 1, 1 <= 2 (True)

Print 1 c = 1 x (2-1) / 1 = 1

j = 2, 2 <= 2 (True)

Print 1

3rd iteration (i = 3)

```
1
1 1
1 2
```

j = 1, 1 <= 3 (True)

Print 1 c = 1 x (3-1) / 1 = 2

j = 2, 2 <= 3 (True)

Print 2 c = 1 x (3-1) / 2 = 1

j = 3, 3 <= 3 (True)

Print 1 c = 1 x (0) / 4 = 0

j = 4, 4 <= 3 (False)