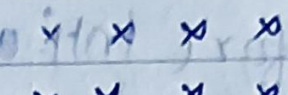
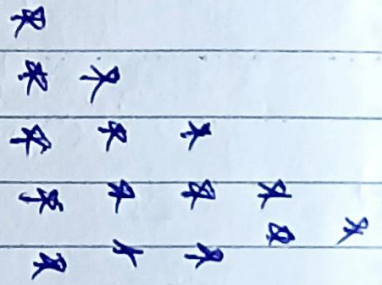
Patterns

1. outer loop, count no. of rows
2. inner loop, focus on columns and connect them with rows
3. print then '*' inside the inner for loop.



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



```
for ( i=0; i<5 ; i++)
  {
    for ( j=0; j<=i ; j++)
      { cout << '*';
      }
    cout<<endl;
  }
```

```
for( int i=1, i<=5, i++)
{ for( int j=1, j<=i, j++)
  { cout<<j;
  }
  cout<<endl;
}
```

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

for every row column are going from 1 to row itself

```
for (int i=1, i<=5, i++)
{
  for( j=1, j<=i, j++)
  { cout<<i;
  }
  cout<<endl;
}
```

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

for every row print row no.

```
for( int i=0 ; i<5 ; i++)
 { for( j=0; j<n-i+1 ; j++)
   {cout << "* ";
   }
  cout<< endl;
 }
```

```
* * * * *
* * * *
* * *
* *
*
```

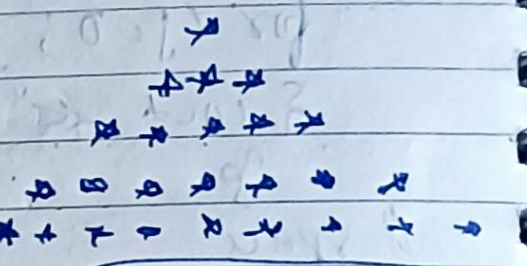
(n-row+1)

1 row → 5 stars
2 rows → 4 stars
3 rows → 3 stars
4 row → 2 stars
5 row → 1 stars

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

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

```
for (i=0; i<n ; i++)
  { //space
    for (int j=0; j<n-i-1; j++)
    { cout << " ";
    }
//star
    for (int j=0 ; j<2*i+1 ; j++)
    { cout << "*" ;
    }
//space
    for (int j=0 ; j<n-1-1 ; j++)
    { cout << " ";
    }
    cout << endl;
  }
```



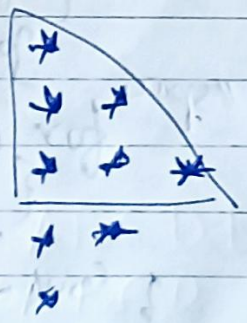
9 columns
[4spau, 1star, 4spau]
[3, 3, 3]
[2, 5, 2]
[1, 7, 1]
[0, 9, 0]

spau. $(n-i-1)$
* $(2i+1)$

(2*n - 2*i + 1)

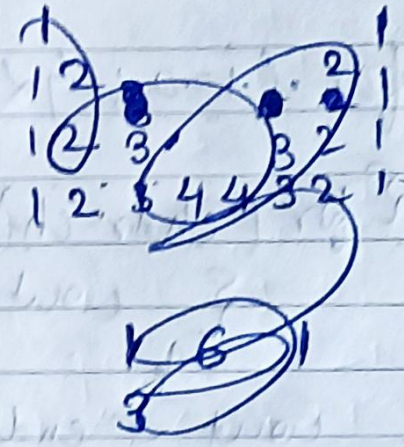
```
for( i=0 ; i< n ; i++)
 { //spaces
   for (j=0; j<i ; j++)
   { cout<< " ";
   }
//stars
   for (j=0; j< 2n-2i+1 ; j++)
   { cout <<"*" ;
   }
// spaces
   for (j=0 ; j<i ; j++)
   { cout << " " ;
   }
   cout<<endl;
}
```

```
for (int i=1; i<2n-1; i++)
 { int stars = i;
   if (i>n) stars = 2*n - i;
   for (int j=1 ; j<stars ; j++)
   { cout << "*";
   }
   cout<<endl;
}
```



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

```
1
0 1
1 0 1
0 1 0 1
1 0 1 0 1
```



```
space = 2*(n-1)
for(i=1 ; i<n ;i++)
{ // number
   for(j=1; j<=n ;j++)
     { cout<<j
     }
  // spac
   for(j=1 ; j<space ; j++)
     { cout<< " ";
     }
  // number
   for(j=i ; j>=1 ; j--)
     cout <<j;
  }
  cout<<endl;
  space -=2;
}
```

```
1 . . . . . 1
1 2 . . . 2 1
1 2 3 . 3 2 1
1 2 3 4 4 3 2 1
```

```
1 6 1
2 4 2
3 2 3
4 0 4
```

```
num = 1
for(int i=1 ; i<=n ; i++)
 {  for (int j=1 ; j<=i ; j++)
    {
      cout << num;
    } num++;
    cout << endl;
 }
```

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

```
for (i=0; j<n ; i++)
{ for ( char ch='A' ; ch <= 'A' + i ; ch++)
    { cout << ch ;
    }
  cout << endl;
}
```

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

```
for (i=0 ; i<n ; i++)
 { for (char ch='A' ; ch<='A'+n-i-1 ; ch++)
    {cout<<ch;
    }
  cout<<endl;
}
```

```
ABCDE
ABCD
ABC
AB
A
```

- ABCDE, A + 4, n=4
- ABC, A+2, n=3
- ABCD, A+3, (n-i-1)

```
for(int i = 0 ; i<n ; i++)
{ for(ch='A' ; ch<='A'+(n-1) ; ch++)
{ char ch = 'A' + i;
    for (int j=0 ; j<i ; j++)
    { cout <<ch << " ";
    }
    cout <<endl;
}
```

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

```
for(int i=0 ; i<n ; i++)
{   //space
    for(int j=0; j<n-i-1 ; j++)
    { cout << " ";
    }
    ch='A';
    int breakpoint = (2*i+1)/2;
//char for(int j=1; j<=2*i+1 ; j++)
    { cout<<ch;
      if (j<=breakpoint) ch++;
      else
      ch--;
    }
//space for(int j=0 ; j< n-i-1 ; j++)
    { cout << " ";
    }
} cout<<endl;
```

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

space n-i-1

char 2*i+1

A B C B A

5/2+1 → int ch='A'

breakpoint

```
for (int i=0 ; i<n; i++)
{ for (char ch='E'-i ; ch<='E'; ch++)
    { cout<<endl;
    }
}
```

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

0 → E E-i
1 → D E

Pairs → part of utility library

pair<int, int> = p = {1, 3};

To access

p.first, p.second

Vectors → container which is dynamic in nature

Array - arr[] = {1, 2, 3, 4} → we cannot modify size of array

- we can always res size of vector whenever you wish to.

vector<int> v; ⟶ it will contains empty container. { }

v.push_back[1];

v.emplace_back[2];

2 ways to push element in vector

push_back

emplace_back → faster

vector<int> v(5); → {0, 0, 0, 0, 0}

vector<int> v(5, 100); → {100, 100, 100, 100, 100}

vector<int> v1(v2);

```
for (vector<int>::iterator it = v.begin(); it != v.end(); it++){
    cout << *it << " ";
}
```

Or.

↗ automaticly assign datatype

```
for (auto it = v.begin(); it != v.end(); it++) {
    cout << *it << " ";
}
```

or using for each loop.

```
for(auto it : v) {
    cout << it << " ";
}
```

[10, 20, 30, 40]

vec.erase (st, end)
↳ end address after the element

vec.insert (v.begin() + 1, 2, 10)
↓
occurance

o/p: {10, 10, 10, 20, 30, 40}

- find() → searches first occurance of a value in a range
  ↳ searches specific element within container and return iterator to the found element

Iterator
↳ points the address.
can access value using *

(LIFO)

Stack.

```
stack<int> st;
st.push(1);
st.pop(2);
st.top();
```

(FIFO)

Queue

```
queue<int> q;
q.push(1);
q.pop(2);
q.front()
q.back()
```

Priority queue

largest value stays at the top.

push, pop, top.

set → sorted, unique

Multiset → sorted, (duplication allowed)

Unordered → unique.

Map → unique keys in sorted order

```
map<int, int> mpp;
```

(key, value → of any datatype)

MultiMap → duplicat keys, sorted

{1,2}, {1,3}

Unordered Map → unique key, not sorted

↙ works in constant time O(1)

Extra sort(a, a+n)

start ↑ ↑ end

automatically sort in ascending

[1, 2, 3, 4, 5]

a ↑ ↑ a+n

Basic Maths Concept

extraction of digit

7789 % 10 = 9

/10 → 778 % 10 = 8

/10 → 77 % 10 = 7

/10 → 7 % 10 = 7

/10 → 0

```
778
10|7789
  7780
  ---
     9
```

Division
/ → return quotient

Floor Division
// → return largest integer ≤ to the quotient
↳ round off to nearest integer

Modulo
% → returns remainder

```
while (N > 0)
{
    last digit = N % 10
    N = N / 10
}
```

↳ Time complexity $O(\log_{10}(N))$

count of digits

$\log_{10}(7789) = 3.89$

+1

4.89 → (take int)

```
#include<bits/stdc++.h>
int count(int n){
    int cnt = (int)(log10(n)+1);
    return cnt;
}
```

Palindrom:

reverse no. and check condition

1331 → reverse 1331

Armstrong:

$371 = 3^3 + 7^3 + 1^3 = 371$

$1634 = 1^3 + 6^3 + 3^3 + 4^3 = 1634$

```
dup = N
sum = 0
while(N>0)
{
    lastdigit = N%10;
    sum = sum + (lastdigit * lastdigit * lastdigit);
    N = N/10
}
```

If (sum == dup) ⇒ armstrong

Print all divisor

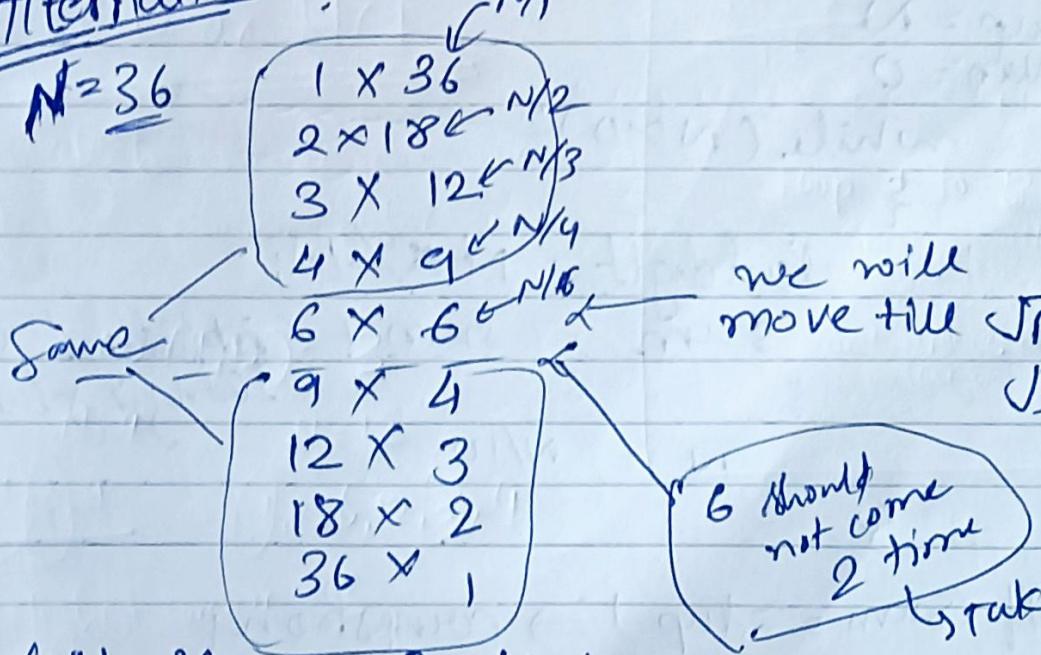
36 → 1, 2, 3, 4, 6, 9, 12, 18, 36

we can say that it lies b/t {1 to N}

```
for (i=1; i<=N; i++)
{
    // completely dividing
    if (N % i == 0)
        print (i)
}
```

Alternative

N = 36



o/p: 1, 36, 2, 18, 3, 12, 4, 9, 6 ← not in sorted manner

```
for (i=1 ; i<=sqrt(N) ; i++)
  {if(N % i==0)
    { print(i)
      if((n/i) != i){
          print(n/i);
    }     }
```

Prime No: → exactly two factor
(1 & itself)

```
counter = 0
for (i=1 ; i<=N ; i++)
  { if (N % i ==0)
      counter++;
  }
if (counter == 2)
    print("Prime")
else
    print("Not prime")
```

Note:
we can also get factor using sqrt(N)

or using sqrt(N)

```
cnt = 0
for( i=1 ; i*i<=n ; i++ )
  { if (n%i ==0)
      { cnt++;
        if (n/i)!=i)
          cnt++;
      }
  }
if (cnt==2)
   Prime
else
   Not Prime
```

→ O(sqrt(N))

GCD or HCF

Greatest common division
Highest common factor

GCD (9,12) = 3

N=9 N=12

1 3 9 1,2,6,12 3,4

Highest common = 3

Logic

let N

N=9 N=12

Loop 1 to min(n1, n2)

Check for all i value where it is divisible by both N1 and N2 and replace it with GCD value

$O(\min(n_1, n_2))$

```
for (i=1 ; i<=min(n1,n2) ; i++)
  { if (n1%i==0 && n2%i==0)
      gcd = i;
  }
```

Euclidean Algorithm → Time complexity $O(\log_{\phi}(\min(a,b)))$

$gcd(a,b) = gcd(a-b, b)$ where $a > b$

$gcd(n_1, n_2) = gcd(n_1 - n_2, n_2)$ $n_1 > n_2$

$gcd(52,10) \rightarrow gcd(42,10) \rightarrow gcd(32,10) \rightarrow gcd(22,10) \rightarrow gcd(12,10) \rightarrow gcd(2,10)$

(1) (2) (3) (4) (5)

$52/10 = 5$
$52\%10 = 2$

$gcd(a,b) = gcd(a\%b, b)$

greater % smaller → one of them zero the other is gcd. $(0, n) \rightarrow gcd$