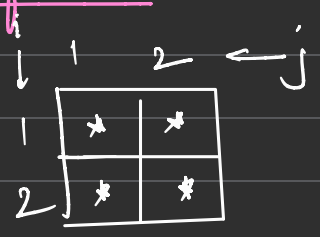
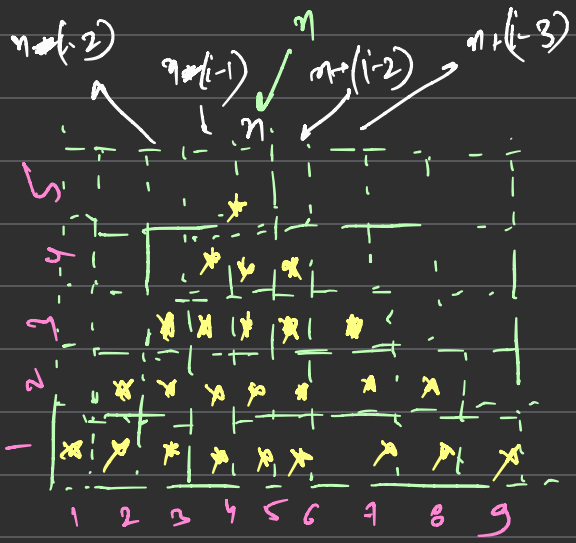
Special Programs in C (Pattern Printing)

Pyramid of Stars :-




```
for(i=1; i<=2; i++)
{
    for(j=1; j<=2; j++)
    {
        printf("*");
    }
}
```



4 × 7

Rows	Columns
4	7
5	9
6	11

If n is the number of rows then 2n-1 will be the number of columns

Code :-

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

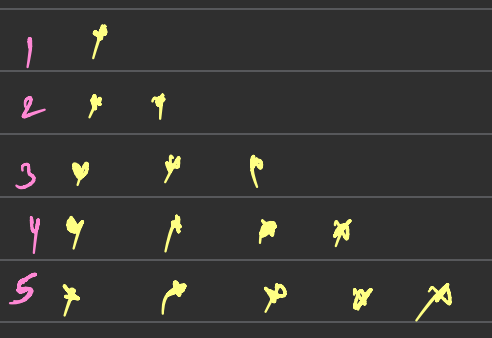
cout << "moving plate n from s to d";

TOH(n, d, s, n-1));

Total number of Steps = $2^n - 1$ (Learn)

PATTERN PRINTING :-

I




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

Steps:-

① Count the number of lines for outer loop

② for inner loop, focus on the columns and connect them somehow to the rows

③ Print the "*" inside other for loop

④ Observe Symmetry

n=5

Ⅰ

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

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

Ⅱ

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

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

Ⅲ

```
    1 2 3 4 5
1   * * * * *   (n-i+1)=5
2   * * * *     (5-2+1)=4
3   * * *       =
4   * *
5   *
```

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

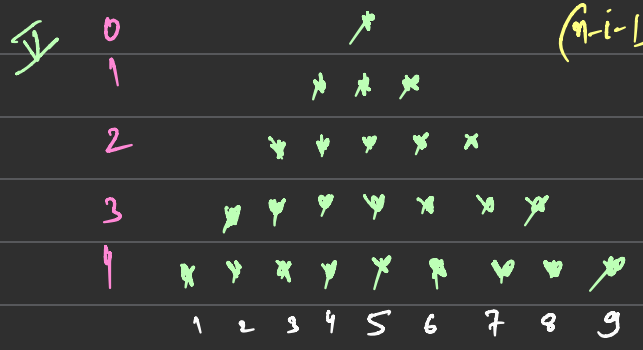
Ⅳ

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

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

space, stars, space

IV



(n-i-1)

4	1	4
3	3	3
2	5	2
1	7	1
0	9	0

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

IV

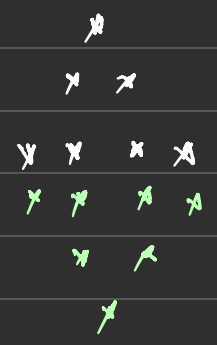
0 → 0, 9, 0
1 → 1, 7, 1
2 → 2, 5, 2
3 → 3, 3, 3
4 → 4, 1, 4

2n − (2i+1) for stars

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

combine functions of V and VI
and you will get the desired output

VII



VIII

n = 5

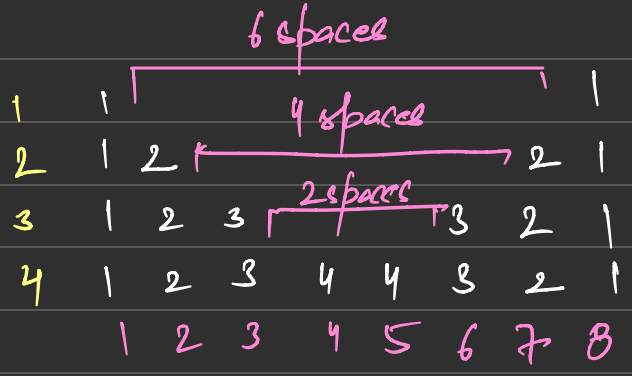
(2n-1) outerloop

```
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;
```

IX

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

```
int start = 1;
 for(int 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;
}
```



outer loop 4 times

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

```
1   A
2   A B
3   A B C
4   A B C D
```

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

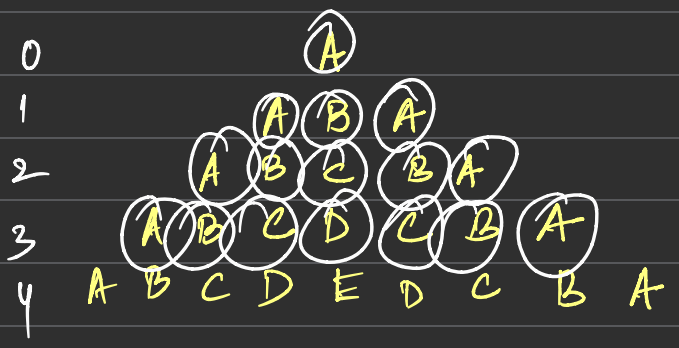
```
A   B   C   D
A   B   C
A   B
A
```

n=4

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

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

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



n-i-1

{ spaces, alphabet, spaces }

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

n-times

```
int breakpoint
    = (2*i+1)/2;
```

```
// characters
char ch = 'A';
for(int j=0; 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;
}
```

E

D E

C D E

B C D E

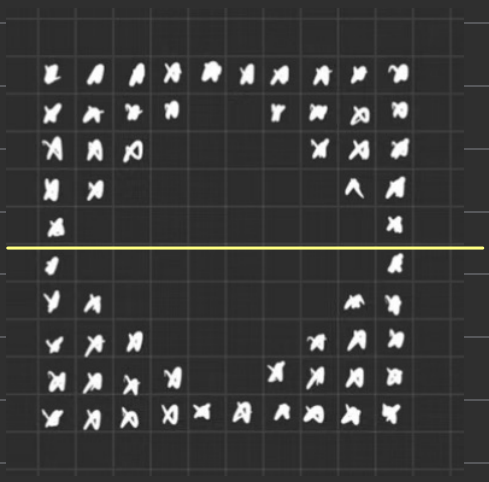
A B C D E

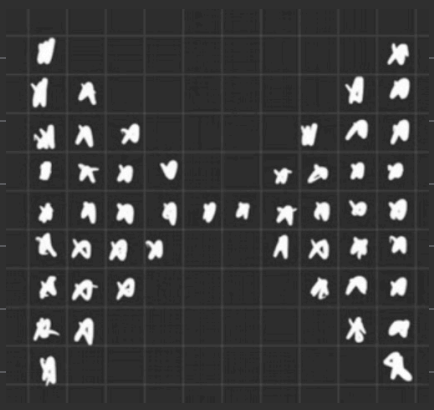
0 → E

1 → D E

2 → C D E

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





	1	2	3	4	5	6	7
1	4	4	4	4	4	4	4
2	4	3	3	3	3	3	4
3	4	3	2	2	2	3	4
4	4	3	2	1	2	3	4
5	4	3	2	2	2	3	4
6	4	3	3	3	3	3	4
7	4	4	4	4	4	4	4

sub every value from n

```
for(int i = 0;i<2*n-1;i++) {
    for(int j = 0;j<2*n-1;j++) {
        int top = i;
        int left = j;
        int right = (2*n - 2) - j;
        int down = (2*n - 2) - i;
        cout << (n - min(min(top, down), min(left, right)));
    }
    cout << endl;
}
```

Number Palindrome or Not :-

Example:- 1221 , racecar, 24542 etc

Idea:- reverse the number and if the reverse number is same, return true

What will happen when we divide the number by 10?

remainder → last number
quotient → remaining number

456/10 → 45 → remaining number
6 → last number (remainder)

Code

result = 0
number = 2332
q = number

remainder = q % 10
result = result * 10 + remainder
q = q/10

```c
#include <stdio.h>

int main() {
    int n, result=0, q, rem;
    printf("Please enter the number: ");
    scanf("%d", &n);

    q = n;

    while(q!=0)
    {
        rem = q%10;
        result = result*10 + rem;
        q = q/10;
    }

    if(result == n)
        printf("Its a palindrome");
    else
        printf("No! its not a palindrome");
    return 0;
}
```

Range of signed int = -2^{N-1} to $2^{N-1}-1$
Where N = No. of bits

Range of unsigned int = 0 to $2^{N-1}+2^{N-1}-1$
= 0 to $2 \times 2^{N-1}-1$
= 0 to (2^N-1)

Armstrong Number

<Q> Check whether a number is Armstrong number or not?

An armstrong number of order n is a number in which each digit when multiplied by itself n number of times and finally added together, results the same number.

eg 371 → 3 digit, order = 3

$3*3*3 + 7*7*7 + 1*1*1 = 27+343+1 = 371$

Code :-

Step 1

```
count = 0;
while( q != 0)
{
    q = q/10;
    count++;
}
```

Step 2

```
ent = count, result = 0;
while( q != 0)
{
    rem = q%10;
    while( ent != 0) {
        mul = mul * rem;
        ent--;
    }
result = result + mul;
ent = count;
q = q/10;
mul = 1;
}
```

ex 1: 371 % 10 = 1
mul = 1
result = 1

2: 37 % 10 = 7
mul = 7*7*7 = 343
result = 1 + 343 = 344

3: 3 % 10 = 3
mul = 3*3*1 = 27
result = 344 + 27 = 371

Write a program to check whether a number is a strong number or not?

strong number is a number in which the sum of factorial of individual digit of a number is equal to the original number.

$145 = 1! + 4! + 5! = 1 + 24 + 120 = 145$

Calculate factorial of each digit and add them

```
q = n, fact = 1, result = 0;

while (q != 0)
{
    rem = q % 10;
    for (int i=0; i<=rem; i++)
    {
        fact = fact * i;
    }
    result = result + fact;
    fact = 1;
    q = q/10;
}
```

Program to check prime number :-

A prime number is divisible by 1 and itself.

9 = 3*3

Part-I finding the sqrt

```
#include <math.h>
int main(){
    int x, int val1;
    val1 = ceil (sqrt(x));
}
```

sqrt calculate square root available in math.h

Part-II Check the Divisibility

```
int val2 = x, count = 0;
for( i=2; i<=val1; i++)
    if(val2 % i == 0)
        count = 1;
}
```

Part 3 Check Number prime or not

```
if ((count==0 && val2 != 1) || val2 == 2 || val2 == 3)
        printf ("%d is a prime number", val2);
else
        printf ("%d is not a prime number", val2);
```

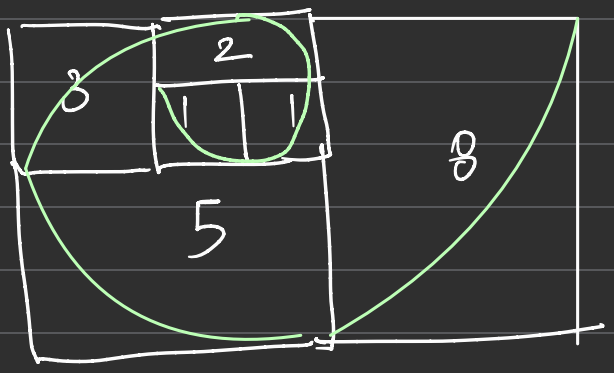
Step-3 Check whether the calculated result is equal to the actual number or not?

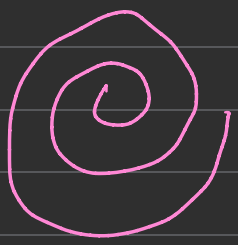
```
if (result == number)
        printf("%d is an Armstrong number", number);
else
        printf("%d is not an Armstrong number",
                number);
```

Add two numbers without using + operator :-

```
int x = 4;
int y = 3;
while( y != 0) {
    x++;
    y--;
}
return x;
```

Fibonacci Number :-





→ Galaxy

Fibonacci Series has lot of practical applications

```
a = 0;
b = 1;

for(int i = 0; i <= n; i++)
{
    printf("%d", a);
    result = a + b;
    a = b;
    b = result;
}
```

Q. What is a floyd's triangle?

A floyd's triangle with 5 rows

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

n = 5

```
int n;
scanf("%d", &n);
int k=1
for(int i=1; i<=n; i++){
    {
    for(int j=1; j<=i; j++){
    printf("%d", n);
    k++;
    }
printf("\n");
}
```

Binary to Decimal Number :-

1 0 0 1

2^3 2^2 2^1 2^0

$1\times2^3 + 0\times2^2 + 0\times2^1 + 2^0\times1$ = Ans

Code :-

```
scanf("%d", binary);
decimal=0, weight=1;
while(binary != 0)
{
    rem = binary % 10;
    decimal = decimal + rem * weight;
    binary = binary / 10;
    weight = weight * 2;
}
```

Power of an Integer :-

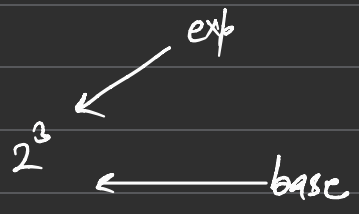
$a^b = \underbrace{a * a * a * a \dots a}_{b \text{ times}}$

Example :-

$2^3 = 2 \times 2 \times 2$

$2^{-3} = 2^{-1} \times 2^{-1} \times 2^{-1}$

Implementation




```
int base, exponent;

int pow = 1, expo;
(stores the result)

expo = exponent;
if (exponent > 0)
{
    while (exponent != 0)
    {
        power = power * base;
        exponent--;
    }
    printf ("%d to the power of %d is %d", base, expo,
                                            power);
}
```

if power is -ve

```
int base, exponent, expo;
    double   power1 = 1.0;
     printf ("Enter the base : ");
     scanf ("%d", &base);
     printf ("Enter the exponent : ");
     scanf ("%d", &exponent);

     expo = exponent;
     if (exponent < 0)
        {
            while (exponent != 0)
              {
                 power1 = power1 * (1.0 / base);
                 exponent++;
              }
             printf("%d to the power of %d is %.1lf",
                 base, expo, power1);
        }
```

Leap Year or Not?

Leap Years has 366 days
extra day is 29th Feb

Note:- Every year is a leap year which is exactly divisible by 4 is a leap year, except the centurial year that is exactly devisible by 100. But these centurial years are leap years if they

if they are exactly divisible by 400.

Code:-

```
int main () {

    int year;
    printf ("Enter the year: ");
    scanf ("%d", &year);

    if (year % 400 == 0)
        printf ("%d is a leap year", year);
    else if (year % 100 == 0)
        printf ("%d is not a leap year", year);
    else if (year % 4 == 0)
        printf ("%d is a leap year", year);
    else
        printf ("%d is not a leap year", year);
    return 0;
}
```

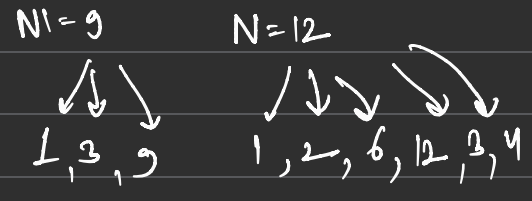
Perfect Number :-

perfect number is a positive integer that is equal to the sum of all its proper positive divisors excluding the number itself.

eg: 6 is a perfect number
Because the proper +ve divisors of 6 are 1, 2 and 3 (excluding 6)

```
printf("Enter the number: ");
scanf("%d", &number);

int i, rem, sum = 0;

for(int i=0; i<number; i++){

    rem = number % i;
    if( rem == 0)
    {
        sum = sum + i;
    }
}

if (sum == number) return true;
else return false;
}
}
```

Greatest Common Divisor / Highest Common Factor



GCD of 11 and 13 = 1

1 to 12

n1 = 9 n2 = 12

more accurate

Code:-

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

T.C - $O(\min(n1, n2))$