Cl-1

Human language		Device language
Alphabet	→	C, C++, Java
Word	→	identifire, keyword Data type
Grammers	→	Rule / statement Syntax

High lavel language → Translators, low lavel.

Human language → Machine language

Source code. → object code.
C, C+, Java → Binary.

When we run code

1. c
2. ....o → Translators.
3. ....exe → for consol bars / input.

Data type

range = $[0 \sim (2^n - 1)]$

*for 3b

$0 \sim 2^3 - 1 = 7$

$0 \sim 7$

Sign bit

0 → +
1 → −

0	0	0	→ 0
0	0	1	→ 1
0	1	0	→ 2
1	0	0	→ −4, 2 sc
1	0	1	→ −3

1st compliment → toggle 0 ⟷ 1

2nd " → " value +1

```
 100
 011
  +1
-----
 100
```

int → 2B, 4B | 1B = 8b

Float → 4B

double → 8B

Char → 1B

Range for n bit,

un sign range $= [0 \sim (2^n - 1)]$

only positive

sign range $= [(-2^{n-1}) \sim (2^{n-1} - 1)]$

Positive, negative

Ex.1 find the range for int,

int $= 2B = 16b$

sign range $= [(-2^{16-1}) \sim (2^{16-1} - 1)]$

$= [(-32768) \sim (32767)]$

Cl-3

Qualifier

memory

int/signed → Positive, negative – 2B, 4B

char/un " → Positive – 2B

Short → 1B

long → 4B

Constant

integer Constant → Decimal (10 base)
→ octal (8 base)
→ hexa decimal (16 base)

Floating Point Constant → Octal start with '0'

Charecters → hexa decimal start with "0x" OR "0X"

Shorting → Zero

Floating point Constant → only work on decimal

1) dot(.) 2) E/e → e=10x

Ex.

1.3e5 → valid

2.3e1.5 → Not valid (Power Can't $10^{1.5}$)

Cl-4
Ch-3

Operators

1 → Arithmetic operators {+, -, *, /, %}
2 → Unary " {-, ++, --, sizeof, (type), !}
3 → Relation " {<, <=, >, >=, ==, !=}
4 → logical " {&&, ||}
5 → Assignment " {&, |, ^, >>, <<, ~}
6 → Conditional " {? :}

① Arithmetic operators. {+, -, *, /, %}

Ex. a = 5, b = 3

a % b = 2
↳ ~~int~~ Dont Work on int. Work on float.

a - b = 2
a * b = 15
a / b = 1 → float = 1.67...

② unary operators: {-, ++, --, sizeof, (type)!}

a++ → increase by 1,
a-- → dicrese by 1

First take a,

a++ { a ; a = a+1
++a { a = a+1 ; a

a-- { a ; a = a-1
--a { a = a-1 ; a

Ex.

```
int a = 5, b = 10, y, x;
x = 50 - b++; // x = 40, b = 11 → Ans
y = ++a - 5; // y = 1, a = 6
x = 40, y = 1, a = 6; b = 11
```

Size of → Variable / data type → memory require..

```
int a = 5, b = 8;
Printf("%d", size of a); // 4
```

to export

type

```
int x, y, sum, avg;
scanf("%d %d %d", &x, &y); // x = 5, y = 2
sum = x + y;
avg = sum / 2; // {x+y / 2 = 5+2 / 2 = 3.5 = 3}
Printf("%d", avg);
```

↳ If we use float

③ Relation operators {>, >=, <, <=, ==, !=}

↳ Not ≠ Equal

use on even or odd number and many other equation.

use on if, else statement for find the maximum or minimum value.

a>b, a>c, b>c

④ logical operator {||, && }

a = 5, b = 6

a > b → F/0
a < b → T/1
} → Return Value, T = 1, F = 0

Computer allways check '0' OR F

OR || → if there one 1, Ans is 1

And && → All 1, Ans 1

Not ! → 1 → 0, 0 → 1

Ex.

a = 6, b = 6, c = 8.

(a > b) || (a < c)

↓ ↓

0 0

Ans → 0

Ex (a >= b) || (a < c)

↓ ↓

0 1

Ans → 1

Ex a = 6, b = 6,

a == b → 1

a != b → 0

a >= b → 1

a <= b → 1

If there is no '0' ~~zero~~ Computer Will take [1/T] by ~~difficult~~ default.

⑤ Assignment operator {=, +=, *=, /=, %.=}

= → are check the ram location.

```
int a;
a = 2
```

a = a+b ⟶ a += b

a += b ⟶ a = a+b

a *=b ⟶ a = a*b

in code: a /=b ⟶ a = a/b

a %=b ⟶ a = a%b → work on int.

⑥ Bitwise {&, |, ^, >>, <<, ~}

① | ⟶ Bitwise - OR - ⊕

& ⟶ " · AND - ⊗

^ ⟶ " XOR - {00 → 0 | same
11 → 0 | value=0

<< ⟶ " left shift

>> ⟶ " Right shift

~ ⟶ " negation (1st. compliment)

Ex. e = a & b || 1

a = 3
b = 5 | e = 1,

Ex

e = a & b || 1

a = 3 → 11
b = 5 → 101

e = 1 → 001

e = a | b || 7

a = 3 → 11
b = 5 → 101

e = 7 → 111

e = a ∧ b || b

a = 3 → 11
b = 5 → 101

e = 6 → 110

e << 1 b = 5 → 00000101

b << 1 || 10 b = 10 → 00001010

increse up to 2x

e >> 1 || 3

b = 5 → 00000101
= 00000010 (2x)

a → Negative a

~a + 1 = −a

~a = −a − 1

a = 3, ~a = −4.

***

Shortcut

x = a << n
x = a × 2^n

<< → increase 2x

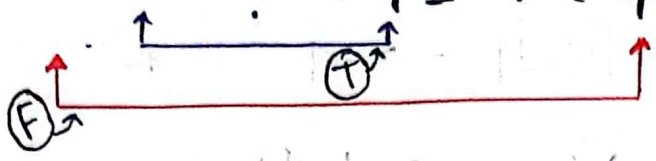
n is input value.

x >> n
x = a / 2^n

>> → dicrese 2x

⑦ Conditional operator { ? : }

Condition ? exp1 : exp2;



True allways first.

Ex. int a=2, b=3, c;
c = a>b ? 5 : 10;
c = 10

int a=2, b=3, c;
c = a<b ? 5 : 10
c = 5

Cl-7
Ch-9

Libraray funtion:

```
# include <stdio.h>
# include <math.h>
# include <string.h>
```

%d → integer
%f → Float fraction (decimal Point)
& → for input.

```
# include <math.h>
```

1) $\sqrt{x}$ = sqrt(x)

i = integer
d = decimal

2) abs(i) = |i| → a=2, b=3, |a-b| = 1

3) exp(d) → exp(2.5) = $e^{2.5}$

4) Fabs (3.5 − 5.9) = |3.5 − 5.9| = 2.4

5) Floor (3.1) = ⌊3.1⌋ → 3 (integers)

6) ceil (3.1) = ⌈3.1⌉ → 4

7) Sqrt (2) = $\sqrt{2}$ → 1.41...

8) log (100) ≐ log 100 → 2

Ex $ax^2+bx+c=0$

$$x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$$

How we write in code

$= x_1 = -b+Sqrt(b*b-4*a*c)$

$x_1 = x_1/(2*a);$

Pow (b, 2) → b × b = b^2

Pow (2, 2) = 2 × 2 = $2^2 = 4$

(2 → base, 2 → Powers)

Scientific eqn:

$10^2 = 10 \times 10$ (or, $10^2 = 10e^2$)

Data input and output

getchar() } → charecters
Putchar() }

scanf() } → All
Printf() }

gets() } → String
Puts() }

Ex. char a; → This allways empty
a = getchar(); || a → They allways taken
Put char (a); First charecters.

When assign

char a = 'ABCD';
a = get char () || ABCD → A → user define
Put char (a); || O → Computer assign They take last value

Scanf Printf

```
scanf("%d", &x);
```

- "%d" → Control string / Format specifire OR string
- &x → Variable

%d → int | & → located memory location
%f → Float | x → memory value.

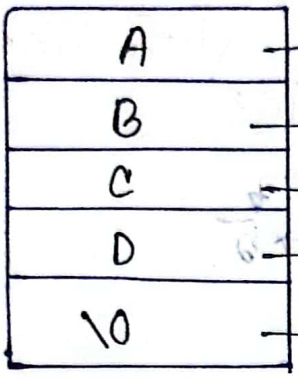
integer (take first alphabate):
- decimal (d)
- Octal (o)
- Hexa decimal (x)
- Float (f)
- charecter (c)
- String (s)
- long (ld)
- long long (lld)
- double (lf)
- short (h)

Ex. String

String

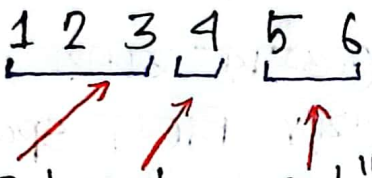
```
Char str[100];
Scanf("%s", &str);  || "ABCD"
```

str [100] store 100 charecter.



{ str start with 0 and end with \0 null

Scanf




```
Scanf("%3d %d %2d", &a, &b, &c);
```

- 'd' mean digit, 2d → 2 digit, 3d → 3 digit.
- %3d → maximum feild width → They take max 3 digit.

Printf

a = 12, b = 256, c = 121

```
Printf("%3d %d %2d", a, b, c);
```

= _12 256 121

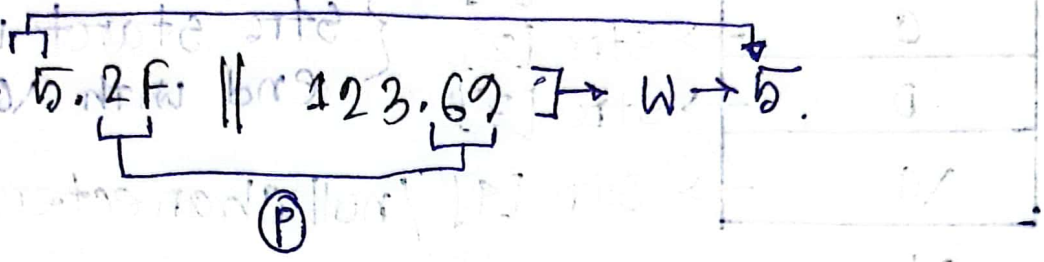
- %3d → minimum feild width → They take minimum 3 digit, space, . → also count

CH-8

Printf

% W.PF

- W → minimum feild width
- P → Presision, How many line he take.
  - * for fixed decimal value



% 3.2F || 1.23 || % 4.2F || 12.34

Scanf এ মান নেওয়ার সময় space এর আগ পর্যন্ত সংখ্যার নিবে, space সহ নিবে,

Ex.

```
int a,b,c ; || 5462012 564 321
Scanf("%2d %d %4d", &a, &b, &c);
// a= 54   b= 62012   c= 564

Printf("%d %d %d", a, b, c);
// 54 62012 564
```

Point also Countable in Float.

Float also add 6 digit by default.

Ex.

```
Char str[100];
scanf("%[AB]", str); // Ab
    // A → output, There AB is capital.
scanf("%[^AB]", str); // abcd → cd
    ^AB → mean they take not (A,B)
scanf("%[^\n]", str); // ABCD EFGH
// ABCD → They show the value before
                of enter or space.
                They take the value before
                of space.
```

^ → Not,

\n → New line.

^\n → Not New line.

```
scanf("%[^\n]", str);
gets(str);
```
} both are same.

```
Printf("%s", str);
Puts(str);
```
} both are same.

Ch-9

* Loop

For loop.

```
for (i = 1 ; i <= n ; i++)
```

- `i = 1` → initialization
- `i <= n` → condition
- `i++` → increment / decrement

Ex.

```
for (i=1 ; i<=n ; i++)
{
    if (i%2==0)        → for even numbers 2, 4, 6 ---
    {                  body
        printf("%d", i);
    }
}
```

input → 10

output → 2, 4, 6, 8, 10

for odd numbers.

```
for (i = 1 ; i<=n ; i++)
{
    if (i%2!=0)        → 2!=0 ~ equal not
    {
        printf("%d", i);
    }
}
```

While.

```
While (i<=5)
```

↳ Condition

→ ~~Condition~~, Continuesly run

The code in the loop will run over and over an long as variable (i) is less than 5.

Ex.

```
While <i<= 5)
{
    Printf("%d", i);
}
```

body

If we dont write |n++| the code will run ∞

Ex. Write a c proagram for print the natural numbers using for & While loop.

for loop.

```
#include <stdio.h>
  int main () {
  int n, i;
 Scanf("%d", &n);
 For (i=1; i<=n; i++)
   {
     if (i  printf("%d\n", i);
   }
 return 0;
}
```

→ Code will be continuously run ~~When~~ an long as variable (i) is ~~less~~ (10)

While loop:

```
# include <stdio.h>
  int main () {
  int n;
  scanf("%d", &n);
  While (n<=10)
  {
    Printf("%d", n);
    n++;
  }
  return 0;
}
```

→ Condition for Value

→ If we dont write increment value, code will run Infinity They dont stop.

DO / While:

```
do {
    Printf("%d\n", i);
    i++;
  }
while (i<5);
```

They always be exicuted at last once.

② → Condition.

Cl-9

if, else

```
# include <stdio.h>          // Max between 2 num
  int main () {
  int  a, b ;
  scanf ("%d %d", &a, &b);
```

a b
a>b — ①
b>a — ②

start with → if (a>b) → Condition

if body:
```
  {
    Printf ("%d", a);
  OR, Printf ("%d a is max:", a);
  }
```

→ end with else, elseif use in middle line

```
else {
    Printf ("%d b is max:", b);
}

return 0;
}
```

1st, if ( ) → Condition
{ }
↳ There is no ;

2nd, else if ( ) → Condition
{ }

end → else → There no condition
{ }

*** Convertion for → while

for

```
# include<stdio.h>
 int main()
 {
  int i,n;
 scanf("%d", &n);
 for (i = 1; i<=10; i++)
 {
  Printf("%d\n", i);
 }
 return 0;
 }
```

while

```
# include <stdio.h>
 int main()
 {
 int i=1, n;
 scanf("%d", &n);
 while (i<=10)
 {
  Printf("%d", i);
 i++;
 }
 return 0;
 }
```

int i = 1 → no Comm

for (i=1; i<=10; i++)

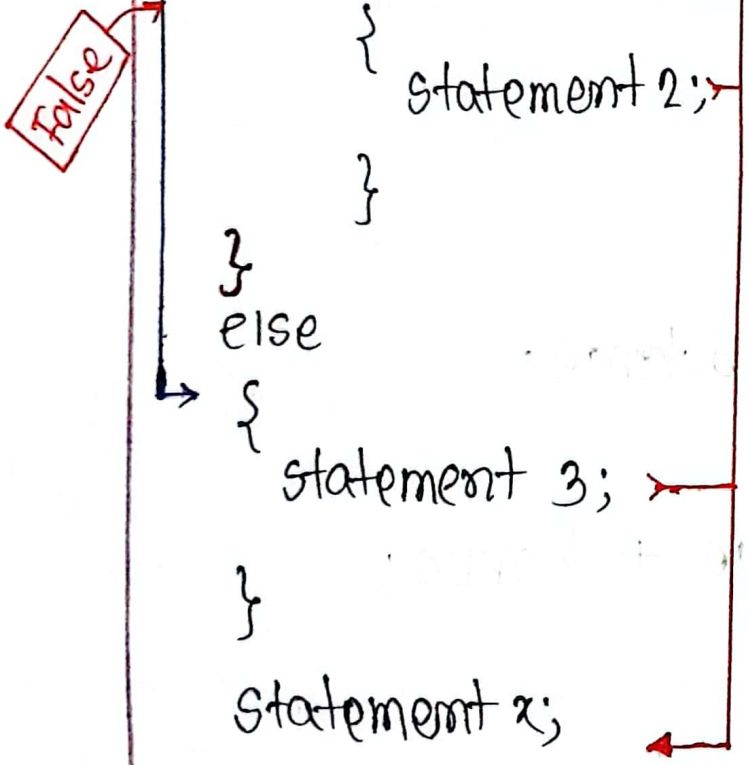
while(i<=10)

after Prin

i++;

Nesting of If - else

```
if (test Condition 1)
  {
    if (test Condition 2);
      {
        statement 1;
      }
      else
      {
        statement 2;
      }
  }
  else
  {
    statement 3;
  }
  statement x;
```



The statement 1 will be evaluated, otherwise the statement 2 will be evaluted

when a series of decision are involved we may have to use more than one If-else statement in nested from

Ex

```
if (sex is female)  ⟶ Condition - 1
{
   if (balance > 5000) → Condition-2
      bonus = 0.05 * balance;
   else
      bonus = 0.02 * balance;
}
else
{
   bonus = 0.02 * balance;
}
balance = balance + bonus;
. . . .
. . . .
```

else - If

```
if (condition 1)
    statement 1;

else if (condition 2)
    statement 2;

else if (condition 3)
    statement 3;

else
    default statement;

statement x;
```

Start the condition with if, end the condition to else statement,

Ex

```
if (code == 1) → Statement 1
  colour = "RED";
else if (code == 2) → statement 2
  colour = "GREEN";
else if (code == 3) → statement 3
  colour = "YELLOW";
else
  No colour; → default
```

Case

switch - Statement.

We have seen that when one of the many alternative is to be selected we can use if statement to control the selection:

```
switch (expression)
{
  case value-1:
        block-1
        break; → to stop the case

  case value-2:
        block-2
        break;

default:
     default-block
        break; → to stop the whole case statement
}

Statement-x;
```

Example of ~~example~~, if-else

Write a c proagrame that take input an a mark of subject and priint the Grade acrcoding to the following.

>=80 → A+ → (m>=80 && m<=100)
>=70 → A → (m>=70 && m<80)
>=60 → B+ → (m>=60 && m<70)
>=50 → C+ → (m>=50 && m<60)
>=40 → D+ → (m>=40 && m<50)
<40 → F → default.

Code

```
#include<stdio.h>
    int main().
{
   float m;
   Scanf("%f", &m);
 if (m>=80 && m<=100)
    Printf("A+");
```

```
else if (m>=70 && m<80)
    Printf ("A");

else if(m>=60 && m<70)
    Printf ("B+");

else if(m>=50 && m<60)
    Printf ("C+");

else if (m>=40 && m<50)
    Printf ("D+");

  else
      Printf ("F");
  return 0;
}
```

Example of switch:

Write a c code that input sub marck and output the grade.

$8 \leq \frac{m}{10} \leq 10 \rightarrow \geq 80 \longrightarrow A+$

$6 \leq \frac{m}{10} < 8 \rightarrow \geq 60 \longrightarrow B+$

$5 \leq \frac{m}{10} < 6 \rightarrow \geq 50 \longrightarrow C+$

$4 \leq \frac{m}{10} < 5 \rightarrow \geq 40 \longrightarrow D+$

$0 \leq \frac{m}{10} < 4 \longrightarrow < 40 \longrightarrow F$

Code

```
#include <stdio.h>
  int main ()
 {
  ~~scanf("%f~~
  float m;
  int i;  ⟶ for float to int value.
 scanf ("%F ", &m);  → input float value
```

```
    i= m/10 ;
switch (i)
{
 case 8:
 case 9:
 case 10
      Printf ("A+");
      break ;
case 6:
case 7:
  Printf(" B+");
     break;
case 5:
     Printf("C+"); → For 50 to 60
     break;
```

→ for 80 to 100

→ break use to stop each statement.

→ for 60 to 70

```
case 4:
    Printf("D+");   } for 40 to 50
    break;

default:
    Printf("F");   } for 0 to 40
    break;
}  ——————→ end for switch.
return 0;
}  ——————→ end for code.
```

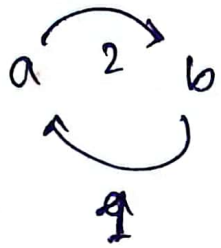
Array

→ array should be used when input value will be used repeatdly.

Write a c programming that take n intger in array and preint the sum of only odd numbers.

input	out
2	13
5	
6	
1	
7	
10	
6	

swap



To exchang the value.

```
#include <stdio.h>
int main()
{
int i, n, a[1000], s;
```

a[1000] → array name; 1000 → array size; i, n → variable name/data type

```
scanf("%d", &n);
 S=0;
for (i=0; i<n; i++)
 {
   scanf("%d", &a[i]);
 }

for(i=0; i<n; i++)
 {
  if (a[i] %.2 == 1)
  st = a[i];
}
 Printf("%d", S);
 return 0;
 }
```

Write a c programm that take n intger in array and store odd and even number in saparate array and Print then,

input → 7, 2, 5, 6, 1, 7, 10, 3

output → even → 2, 6, 10

odd → 5, 1, 7, 3.

```
#include <stdio.h>
 int main ()
 {
 int i,n, o [1000], e [1000], x;
 scanf("%d", &x);
 if (x%2 == 0)
 {
 e[ie++] = x;
 }
  else
      o[io++] = x;
```

for check odd or even

or use

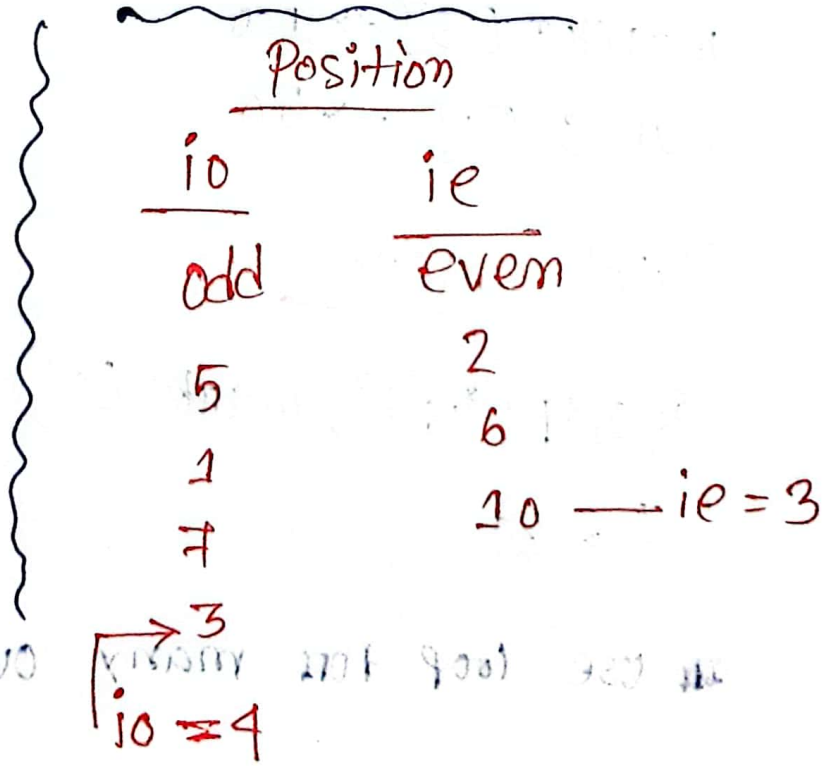
{ Po, Pe → Even, → Odd

for check Position.

```
Printf("even:\n");
for (i=0; i<ie; i++)
{
  Printf("%d\n", e[i]);
}
Printf("%.odd:\n");
for (i=0; i<io; i++)
{
  Printf("%d\n", o[i]);
}
return 0;
}
```

2, 6, 10 // ie = 3

5, 1, 7, 3 // io = 4



Array

int mark [5]
↓ └→ array name
datatype
└→ [5] array size

Array is a Collection of Variable of Same type.

int mark [5] → mark[0], mark[1] ....

array initialization.

marks [0] → 80 mark [0] → Index.
marks [1] → 70
marks [2] → 60
marks [3] → 100
marks [4] → 50

Print array

Printf ("%d", marks [0]);
└→ 80

Use loop for many output.

Matrix

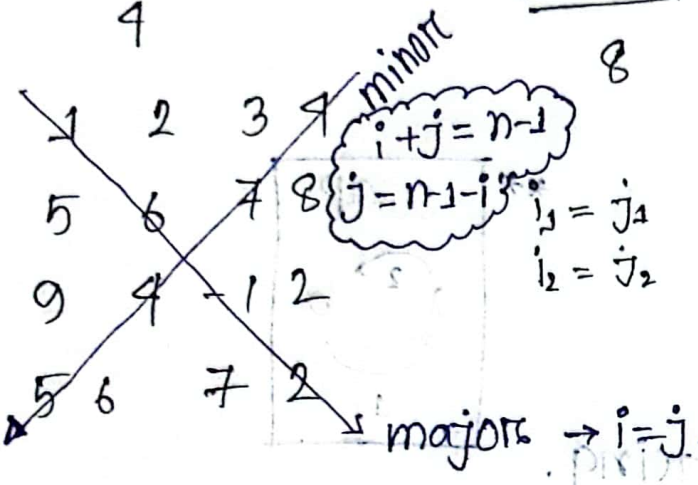
Write a C code that find the sum of major diagonal of n matrix

input

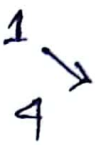
4

output

8



Diagonal when the matrix is square.



square.

i = j

```
#include <stdio.h>
int main ()
{
int i,j,a [100] [100], s, n;
scanf ("%d", &n);
for (i=0; i<n; i++)
 {
   for (j=0; j<n; j++)
        scanf ("%d", &a [i] [j]);
 }
 s=0;
for (i=0; i<n; i++)
 {
```

* major

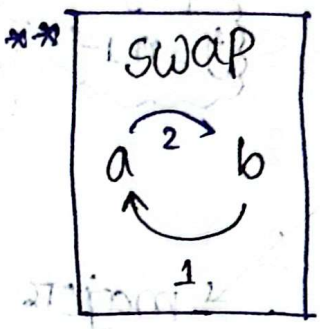
i = j

* minor

j = n-1-i

or, i+j = n-1

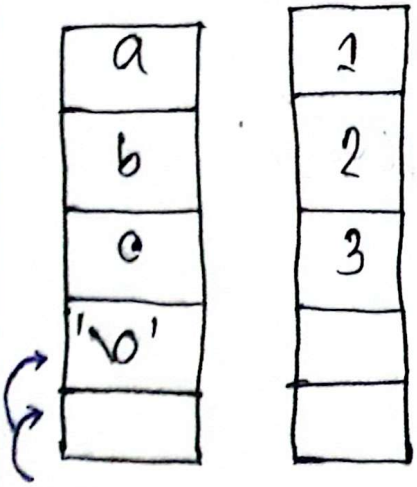
```
s += a[i][i]; //s+a=[i][n-i-1]
                 i=j. so→a[i][i]
}
 Printf("%d", s);
return 0;
}
```



String.

Collection of characters is called string.

`s[5]="abc";` `a[5]={1,2,3};`



space is valid in string

* '\0' is mean stop the string. null default assign by compilors.

funtion

```
Strc len (str 1)
Strc cpy (str 1, str 2)
Strc cmp (str 1, str 2)
Str  cat (str 1, str 2)
```

Ex.

```
Char str[100];
scanf("%s", str); // md hanan \n
```

%[^\n] → Not New line.

* s[5] = "a b c"
  s[5] = {'a', 'b', 'c'}
  } both are same and valid.

"....." → String. double Quotation

'...' → Character. single Quotation

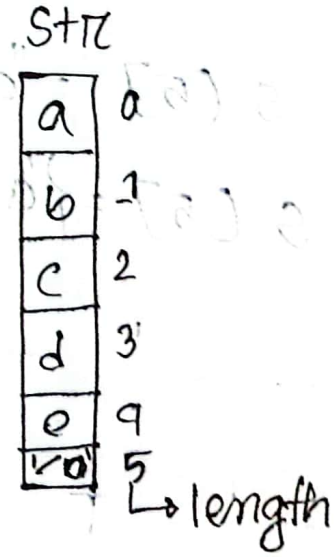
Input 5. Output

5

abcde abcde

→ int n;
char str[1000];
scanf("%d", &n); // 5
scanf("%s", str); // abcde
Printf("%s", str);

Ex. length of string.



int L;
char str[1000];
scanf("%s", str);
for(i=0; str[i]!='\0'; i++)

↳ check the null character '\0'
This is the length of string. where the '\0' is end.