Algorithm: An algorithm is a step by step procedure to solve a problem. (algorithm cannot be executed)

For Writing the Algorithm few points are to be remember. They are

* All the steps we Write in algorithm must have a proper meaning i.e. unnessary steps are to be avoided.
* An algorithm Will start must be completed. i.e the algorithm should not be infinite.

Types of Algorithm:

There are three types of algorithm. They are

1) Sequence algorithm
2) Conditional / decision making algorithm
3) Looping / Iterative

Sequence Algorithm: In sequence type of algorithms all the algorithms Contains Input, Calculation, Output. There is no change of any Condition Checking or Repetation of Steps.

Conditional Algorithm: In Conditional type of algorithms definitly there Will be a decision making Step.

Looping Algorithm: In looping algorithm Certain Steps gets executed Repeatedly until the Condition gets failed.

Write an algorithm for addition of two numbers?

1, Start
2, read a, b values
3, Calculate c = a+b
4, Display C
5, Stop

Write an algorithm to read initial velocity (u), acceleration (a), final velocity (v), time. Calculate Final velocity.

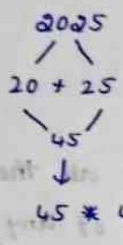
1, Start
2, read u, a, t values
3, Calculate v = u * a * t (v = u+at)
4, Display v
5, Stop

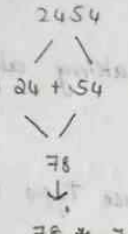
Write an algorithm to find the Simple Interest (SI). Read the Principal amount (P), Time (T) and Rate of Interest (R).

1, Start
2, Read P, T, R values
3, Calculate SI = (P*T*R)/100
4, Display SI
5, Stop

% 10 ⇒ gives last digit
% 100 ⇒ gives last 2 digit
% 1000 ⇒ gives last 3 digit
/ → Quotient
% → remainder
x = 125 % 10 ⇒ 5
y = 125 / 10 ⇒ 12

Write an algorithm to read a 4 digit number. Evaluate that in a following manner.





1, Start
2, Read n
3, Calculate
x = n/100 → division
y = n % 100 → remainder
4, Calculate
z = x + y
5, display z * z
6, Stop

1, Start
2, Read n
3, calculate
x = n/100
y = n%100
4, calculate
z = x + y
5, display z * z
6 Stop

Flow Chart: pictorial or diagrametical representation of an algorithm is a flow chart. There are certain symbols to be used in flow chart.

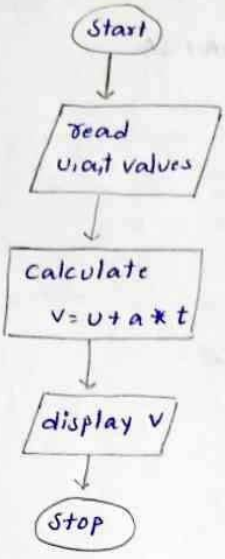
Oval — ⬭ — start/stop

Parallelogram — ▱ — Input/output

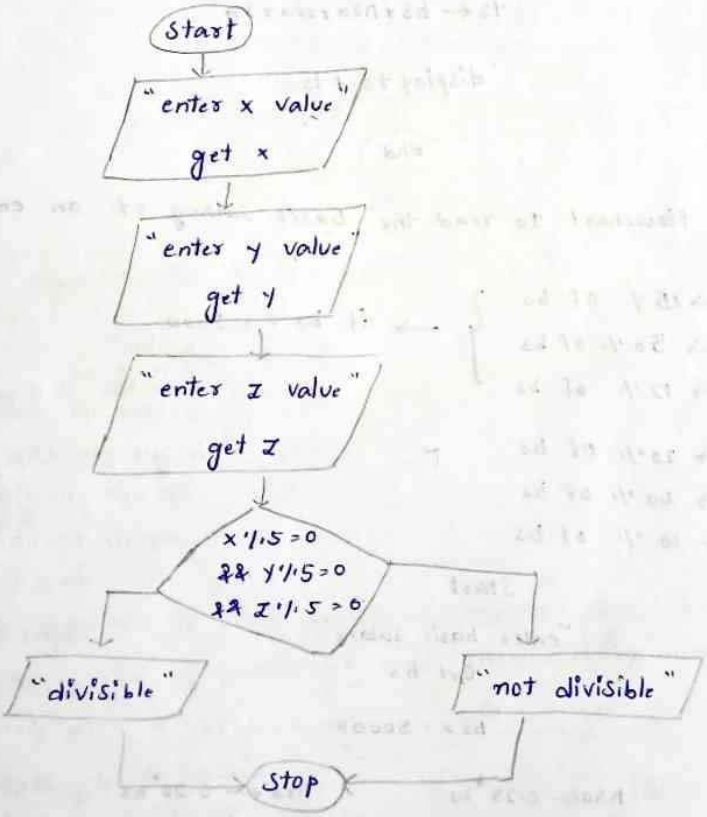
Rectangle — ▭ — computations/calculations

Rhombus — ◇ — Condition checking

Draw the flowchart for v = u + at



Draw a flow chart to read 3 numbers i, y, c & check all the 3 numbers are divisible by 5?



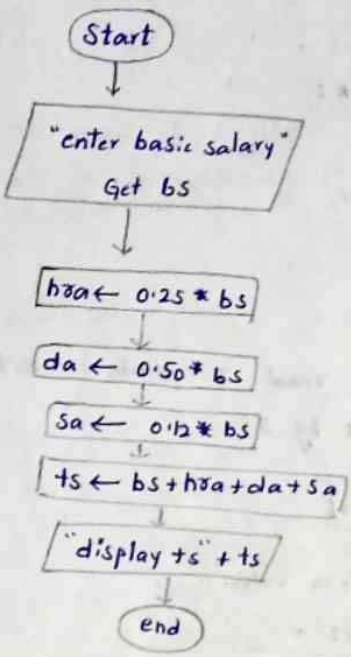
Draw the flowchart to read the basic salary of an employee and Calculate the final salary of employee.

Final Salary = Basic salary + HRA + DA + SA

HRA is 25% of Basic salary

DA is 50% of Basic salary

SA is 12% of Basic salary



Draw the flowchart to read the basic salary of an employee and Calculate

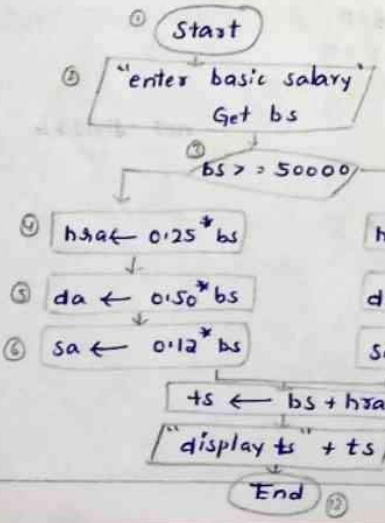
HRA → 25% of bs
da → 50% of bs
SA → 12% of bs

} → if bs >= 5000

other wise

HRA → 20% of bs
da → 40% of bs
Sa → 10% of bs

1) Start
2) read bs
3) check the condition if bs >= 50000
if yes goto step 4
other wise goto step 7
4) Calculate hra = 0.25 * bs
5) " da = 0.5 * bs
6) " sa = 0.12 * bs goto step 10
7) " hra = 0.2 * bs
8) " da = 0.5 * bs
9) sa = 0.1 * bs
10) Calculate ts
ts = hra + da + sa + bs
11) display ts
12) Stop



Write an algorithm & flowchart to read a 4 digit number. Evaluate in the following mannar.

2025

20 + 25

45

45 * 45 = given no

if yes ⇒ hai

no ⇒ bai

1) Start

2) Read n

3) Calculate

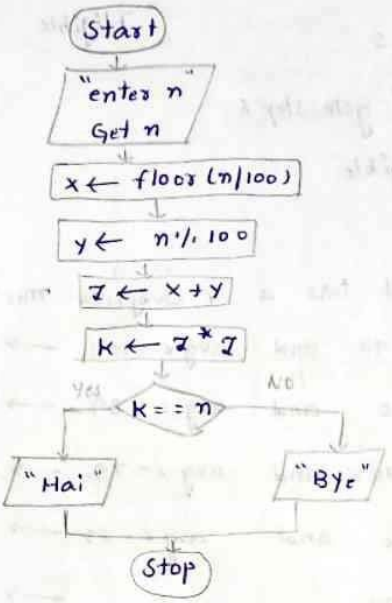
$x = n/100$

$y = n \% 100$

$z = x + y$

4) $z * z$

5) Stop



Write an algorithm for $s = ut + \frac{1}{2}at^2$

1) Start

2) Read u, t, a values

3) Calculate $s = ut + \frac{1}{2}at^2$

4) display s

5) Stop

Write an algorithm to read 6 subject marks. Calculate sum of 6 subjects then find its average and then display sum, avg

1) Start

2) read a, b, c, d, e, f values

3) Calculate sum = a+b+c+d+e+f

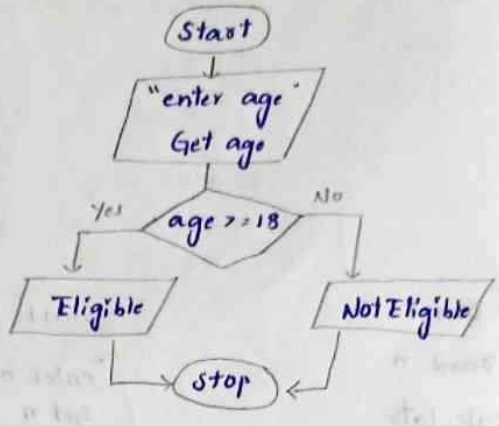
4) Calculate avg = $\frac{sum}{6}$

5) display sum, avg

6) Stop

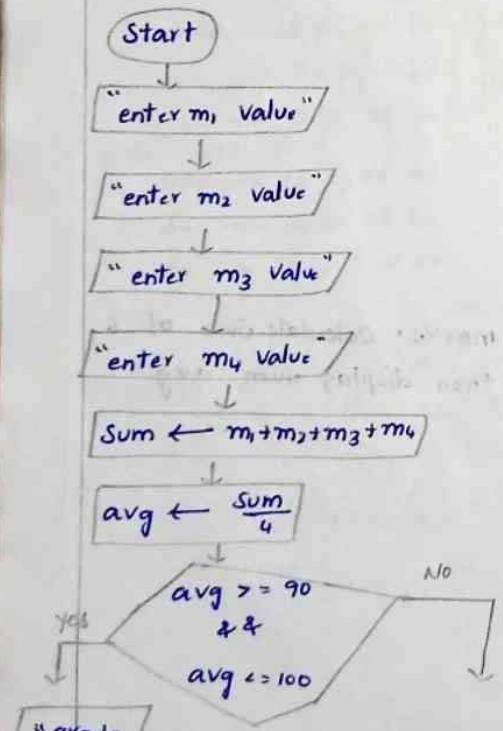
Write an algorithm and flowchart to check whether the person's age is eligible for voting or not.

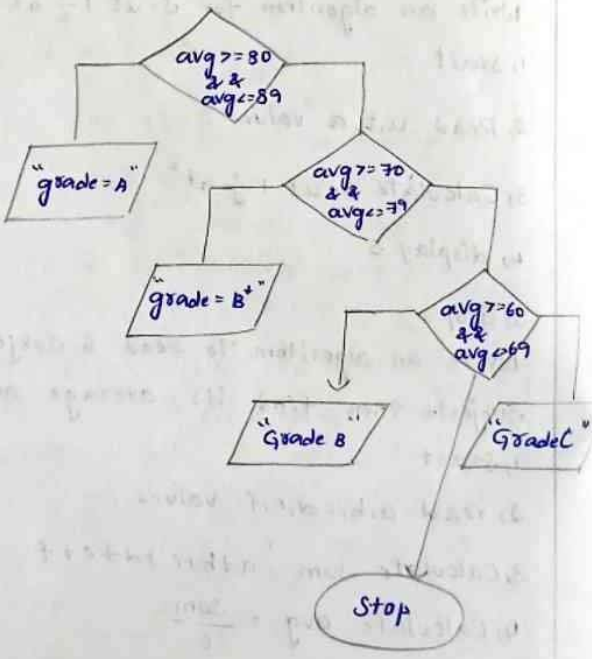
1) Start
2) read age
3) check the condition
if age >= 18
if yes goto step 4
otherwise goto step 5
4) display "eligible" goto step 6
5) display "not eligible"
6) Stop



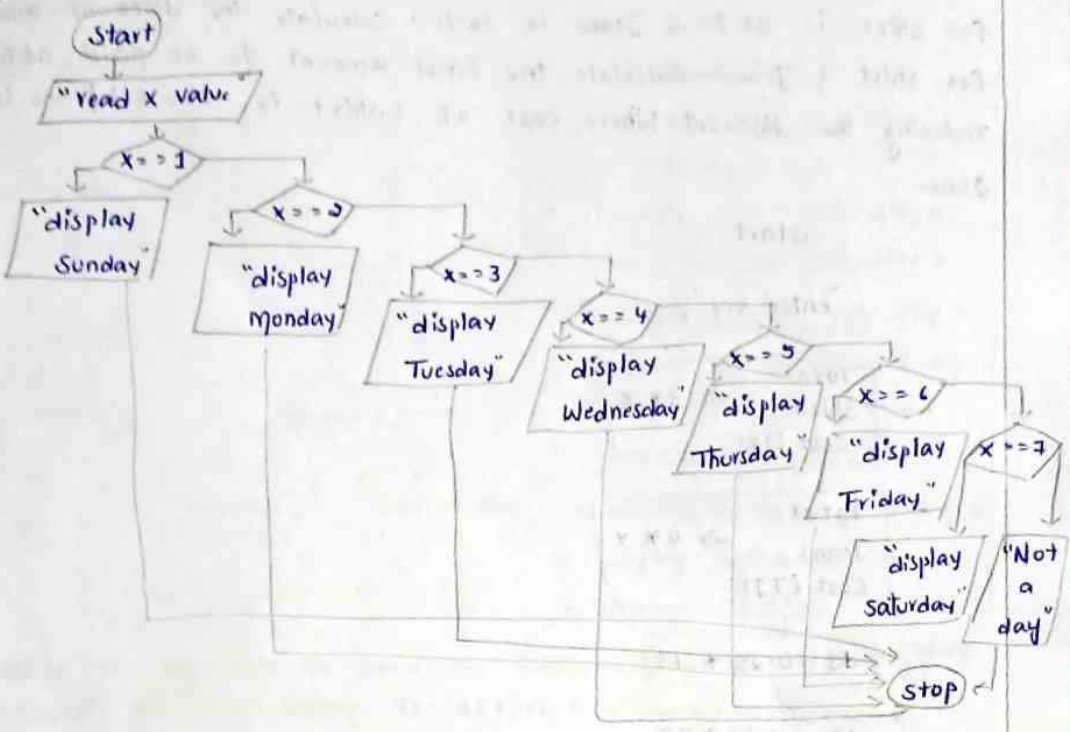
In a College student has a 4 subjects m_1, m_2, m_3, m_4. Find out the avg if

avg >= 90 and avg <= 100 → A+
avg >= 80 and avg <= 89 → A
avg >= 70 and avg <= 79 → B+
avg >= 60 and avg <= 69 → B
avg <= 59 → C

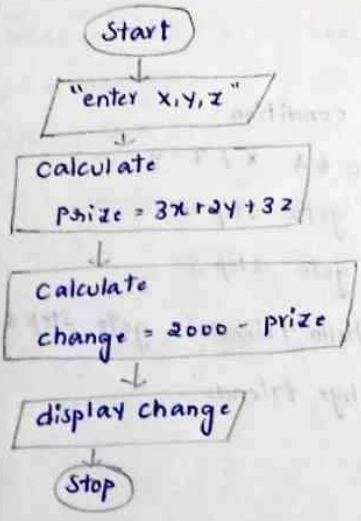




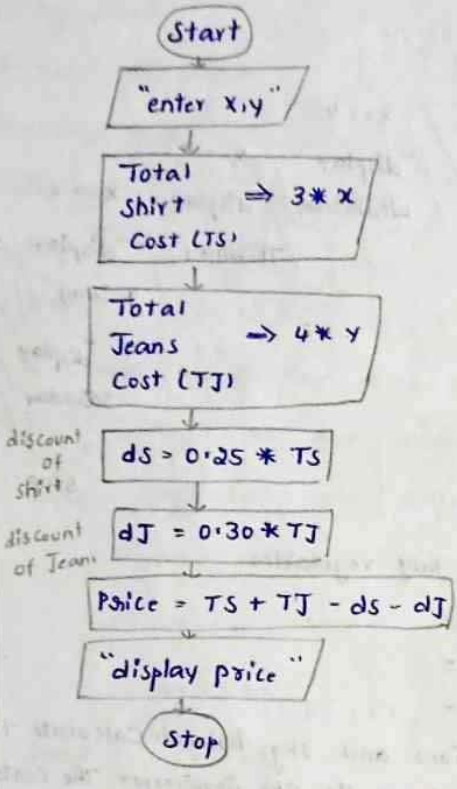
Write a flowchart · To print input value 1 = Sunday ; 0 = monday ; 3 = Tuesday , 4 = Wednesday , 5 = Thursday , 6 = Friday , 7 = saturday.



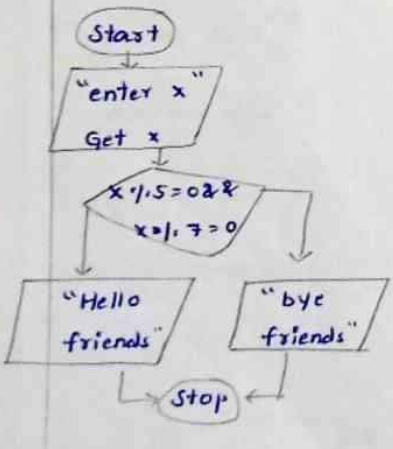
A customer goes to the shopkeeper to buy vegetables.
The price of tomato per 1kg is 85/-
The price of onion per 1kg is 64/-
The price of Brinjal per 1kg is 48/-
customer buys 3kg tomatoes, 2kg onions and 3kg brinjal. Calculate the total amount to be paid by the customer to the shopkeeper. The customer gave 2000/- note to the shopkeeper. Find the total change he gets.



Develop the flow chart to read the price of the shirt, jeans & calculate the total price for 3 shirts & 4 jeans. The discount for shirt is 25% & Jeans is 30%. Calculate the discount amount for shirt & Jeans. Calculate the final amount to be paid after reducing the discount. Where cost of 1 shirt is 1500 & 1 Jeans is 2000.

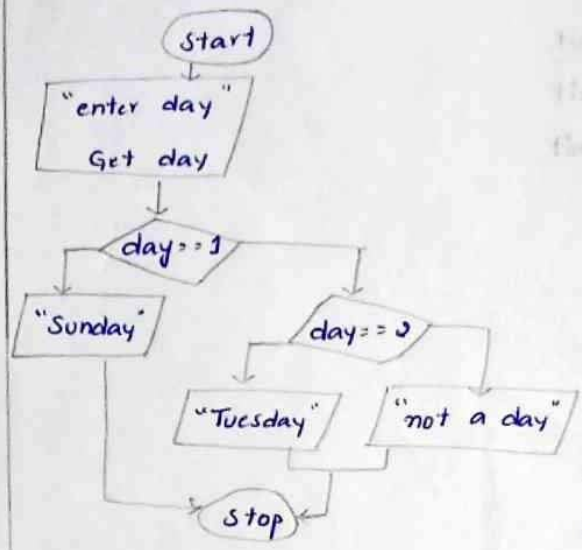


Write an algorithm to read the value and check it is divisible by both 5 and 7. If else display "Hello friends" else display "bye friends".



1, Start
2, read x
3, Check the condition
if x%5 = 0 && x%7 = 0
if yes goto step 4
else goto step 5
4, display "hello friends", goto Step 6
5, display "bye friends"
6, Stop

Write an algorithm to read a day if it is one then display Sunday. Other Wise if it is 2 display monday. otherwise display not a day.

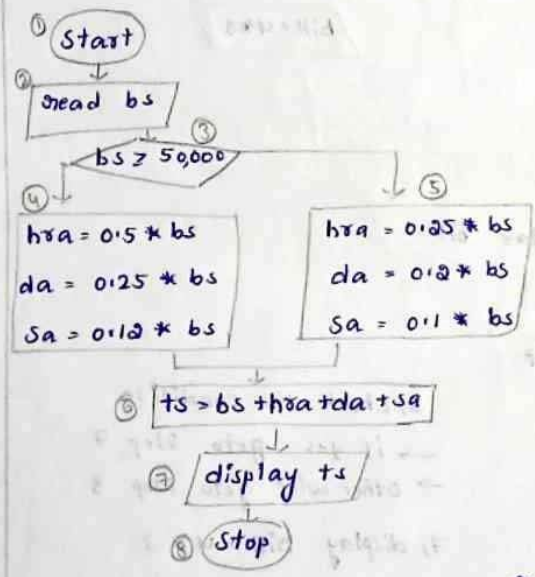


1) Start
2) Read day
3) Check the Condition
   if day == 1
   → if yes goto Step 4
   → otherwise goto Step 5
4) display Sunday goto step 8
5) Check the Condition if day == 2
   → if yes goto step 6
   → otherwise goto step 7
6) display monday goto step 8
7) display not a day
8) Step

Write an algorithm to read the basic salary of an employee & calculate the total salary. Ts = bs + hra + da + sa. Consider.

hra = 50% of bs
da = 25% of bs
sa = 12% of bs
} only if bs >= 50,000

Other wise Consider
hra = 25% of bs
da = 20% of bs
sa = 10% of bs

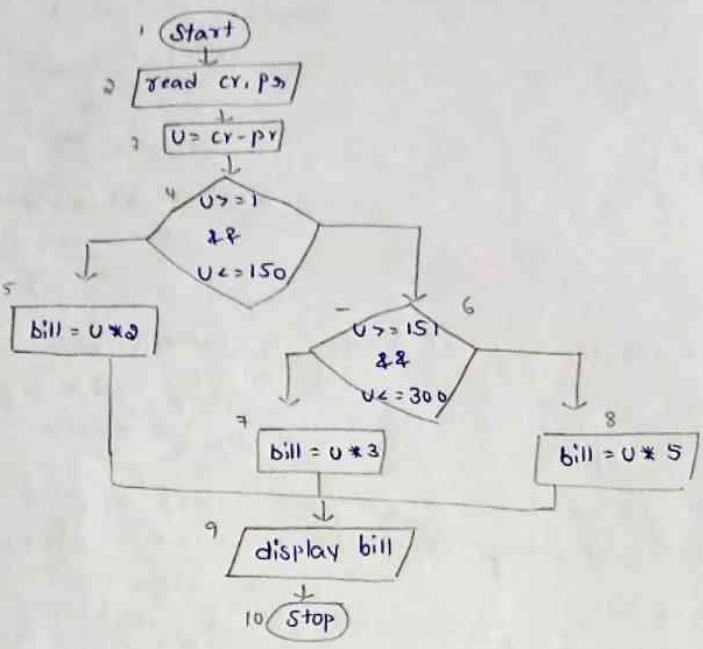


1) Start
2) read bs
3) check the condition if bs >= 50,000
   if yes goto step 4
   Otherwise goto step 5
4) Calculate hra = 0.5 * bs
   da = 0.25 * bs
   sa = 0.12 * bs goto step 6
5) Calculate hra = 0.25 * bs
   da = 0.2 * bs
   sa = 0.1 * bs
6) Calculate ts = hra + da + sa + bs
7) display ts
8) Stop

Write an algorith & flowchart for the following cenario. Read cr, pr,

no. of units consumed (u) = cr - pr

Units	price
1-150	2/unit
151-300	3/unit
other than this	5/unit



1, Start

2, Read cr, pr

3, calculate U = cr - pr

4, Check the condition

if U >= 1 && U <= 150

if yes goto step 5

otherwise goto step 6

5, Calculate bill = U * 2 goto step 9

6, Check the condition

if U >= 151 && U <= 300

if yes goto step 7

other wise goto step 8

7, Calculate bill = U * 3 goto Step 9

8, Calculate bill = U * 5

9, Display Bill

10, Stop

Write an algorithm and draw the flow chart to read the annual income of an employee based on the following criteria. Calculate inhand salary of an employee annually after tax deduction.

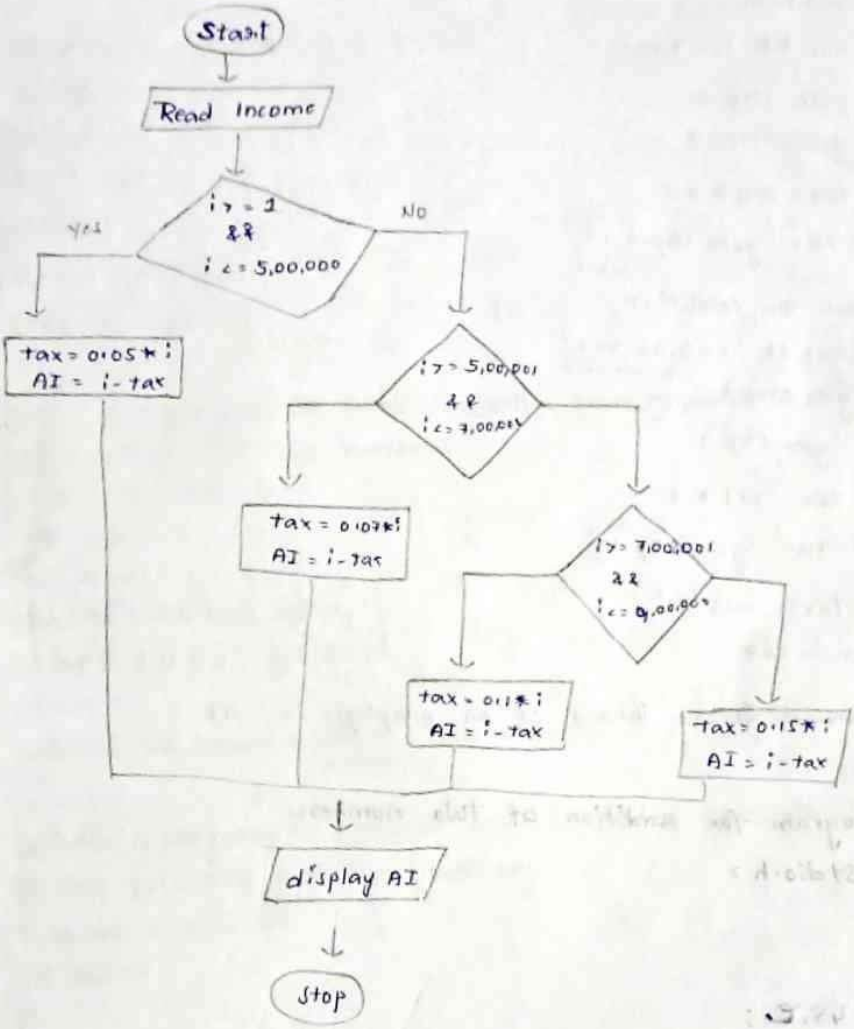
Annual income < 5,00,000

Tax = 5%.

5,00,001 - 7,00,000 ⟶ Tax = 7%.

7,00,001 - 9,00,000 ⟶ Tax = 10%.

OtherWise ⟶ Tax = 15%.



1) Start

2) Read income ;

3) Check the condition

; >= 1 && ; <= 5,00,000

if yes go to step 4

otherwise goto step 5

4) Calculate tax = 0.05 * ;

AI = ; - tax goto step 10

5) Check the condition

; >= 5,00,001 && ; <= 7,00,000

if yes goto step 6

otherwise goto step 7

6) Calculate tax = 0.07 * ;

AI = ; - tax goto step 10

7) ~~Calculate~~ Check the condition

; >= 7,00,001 && ; <= 9,00,000

if yes goto step 8

otherwise goto step 9

8) Calculate tax = 0.1 * 1

AI = ; - tax goto step 10

9) calculate tax = 0.15 * ;

AI = ; - tax

10) Display "Annual Income inhand of an employee is", AI

11) Stop

Write a C program for addition of two numbers.

```
#include<stdio.h>
main()
{
int a=10, b=45, c;
c=a+b;
Printf("%d", c);
}
```

Write a c program for $v = u + at$

```
#include <stdio.h>
main()
{
int v,u,a,t;
u=10; a=10; t=10;
v=u+a*t;
Pointf("%d", v);
}
```

#include <stdio.h> ⇒ Headerfile
main(){ ⇒ main function
Global variable declaration
int a,b,c; ⇒ Local variable declaration
Printf
Scanf
}

Write a c program for $A = \pi r^2$.

```
#include <stdio.h>
main()
{
int A,r;
A=3.14 * r * r;
Pointf("%d", A);
}
```

Write a c program to read two values from keyboard & display the result of addition of two numbers.

```
#include <stdio.h>
int main() {
int a,b,c;
Pointf("Enter a,b values");
Scanf("%d%d", &a, &b);
c=a+b;
Pointf("The result of addition of two numbers is %d", c);
}
```

Write a c program to read 5 subject marks. calculate its sum and its avg & display both sum and avg.

```
#include <stdio.h>
int main()
{
int m1, m2, m3, m4, m5, sum, avg;
Pointf("Enter 5 Subject marks);
Scanf("%d %d %d %d %d", &m1, &m2, &m3, &m4, &m5);
Sum = m1+m2+m3+m4+m5;
avg = sum/5;
Pointf("Sum= %d  avg= %d", Sum, avg);
}
```

Structure of C Program

Structure: optional comment section → /* program for addition of 2 numbers */
↓
Compulsory header file section → #include <stdio.h>
↓
optional Global variable declaration
↓
Compulsory main function → main()
{
↓
optional local variable declaration → int a,b,c;
↓
Compulsory Stament section → Printf("enter a,b values");
scanf("%d%d", &a, &b);
c = a+b;
Printf("%d", c);
}

Comment Section:

In C language Comment section can be represented with either /* ----- */ (or) // symbol. Where 1st symbol can be used When we want to comment either one line or more than one line. Where as the second symbol can be used strictly for single line Comment. The commented lines are ignored by the compiler and doesn't involve in execution past.

Header file Section:

It is used for including library files.

Ex: #include <stdio.h> Where # is a preprocessor directive which includes the standard input output header file. like this there are many library files which can be included as per requirement.

Global variable declaration:

This section is used for declaring the variables which can be used throughout the program by any function.

Ex: int a,b

main function:

Execution of C program starts with main function. Without main function it is not possible to execute the program.

local variable declaration:

The variables which are declared within the function are said to be the local variable and the section is said to be

local Variable declaration Section.

Statement Section:

The instructions given to the Computer are considered as statements.

Every statement ends with ; → semicolon.

Control Statements

conditional	Iterative
if	for
if else	while
else-if Ladder	do While
nested if else	
Branching → switch, goto	

* Control statements are used to ~~make~~ take care of execution flow based on the condition.

* Control statements are divided into two types. They are 1, conditional or decision making statements. 2, Iterative or looping statements.

if

if is the keyword used to check the condition. If the condition is true then the statements which are within the if block are executed.

Ex:

```
#include <stdio.h>
main()
{
int a;
Printf("enter a value");
scanf("%d", &a);
if (a>=0)
Printf("positive");
}
```

In the above program the message "positive" gets displayed only if the condition is true. Otherwise, system will not do anything as there is no Proper direction mentioned when the condition failed.

if else

In if else two key words are used if is used to check the condition and execute its statement only if the condition gets true/satisfied. Otherwise, the statements in else block gets executed.

Syntax:

```
if (condition)
{
  Statements;
}
else
{
Statements;
}
```

Write a C program to check whether the person's age is eligible to vote or not.

```
#include <stdio.h>
main()
{
int x;
Printf("enter x value");
Scanf("%d", &x);
if (x>=18)
Printf("eligible to vote");
else
Printf("not eligible to vote");
}
```

Write a C program to read u, a, t, s values and find s value using the formula $s = ut + \frac{1}{2}at^2$. Here output should be in decimal form.

```
#include<stdio.h>
main()
{
float u, a, t, s;
Printf("enter u, a, t, s");
```

```
scanf("%f %f %f %f", &u, &a, &t, &s);
s = u*t + 0.5*a*t*t;
printf("%f", s);  (or) printf("%0.5f", s);
```

this is used to display 5 digits after point.

}

Write a c program to read big integer value & display that value.

```
#include <stdio.h>
main()
{
long int a;
printf("enter a value");
scanf("%ld", &a);
printf("%ld", a);
```

Write a c program to read 3 numbers and all the numbers should be divisible by 5. If yes calculate the sum of 3 numbers other wise display the message not divisible.

```
#include <stdio.h>
main()
{
int x, y, z, s;
printf("enter x, y, z values");
scanf("%d %d %d", &x, &y, &z);
if (x%5 == 0 && y%5 == 0 && z%5 == 0)
{
s = x+y+z;
printf("%d", s);
}
else
{
printf("not divisible");
}
```

Write a program to read two numbers & check the first number whether it is divisible by 5 and check the second number is divisible by 9. If both gets divisible by its respective numbers calculate the addition of two numbers. Otherwise calculate Subtraction.

```
# include <stdio.h>
main()
{
int x,y,z;
Printf("enter x,y values");
Scanf("%d%d",&x,&y);
if (x%5==0 && y%9==0)
{
  z=x+y;
  Printf("%d",z);
}
else
{
  z=x-y;
  Printf("%d",z);
}
}
```

else-if Ladder:

Syntax for else-if Ladder is

```
if (condition1)
{
(statements;
}
else if (Condition2)
{
statements;
}
else if (condition 3)
{
Statements;
}
:
else
{
statements;
}
```

Write a C-program to find biggest of three numbers.

```
#include<stdio.h>
main()
{
int a,b,c;
Printf("enter a,b,c values");
Scanf("%d%d%d",&a,&b,&c);
if(a>b&&a>c)
Printf("a is big");
   else if(b>c)
Printf("b is big");
   else
Printf("c is big");
}
```

Devolp a C program to find biggest of four numbers.

```
#include<stdio.h>
main()
{
int a,b,c,d;
Printf("enter. a,b,c,d values");
Scanf("%d%d%d%d",&a,&b,&c,&d);
if(a>b&&a>c&&a>d)
Printf("a is big");
   else if(b>c&&b>d);
Printf("b is big");
  else if (c>d)
 Printf("c is big");
  else
Printf("d is big");
}
```

Operators: An operator is used to perform certain operation on operands.

Ex: a+b

In the above example '+' is the operator which performs addition operator on operance a,b.

→ There are totally 8 types of operators are these. They are

1, Arithmetic operator.

2, Relational operator

3, Logical Operator

4, Conditional or Tervenary operator

5, Assignmnt operator

6, Increment / Decrement operator

7, Bitwise Operator

8, Special operator

Arithmetic Operator: Arithmetic operators perform arithmetic operations like addition, multiplication, subtraction, division etc.

Operation Symbol	Name	Operation
+	Addition	adds number
-	subtraction	subtracts number
*	multiplication	Multiples numbes
/	division	returns quotient value
%	modulo	returns remainder value

Relational Operator:

Operator Symbol	Name	Operation
= =	equal to	Compares LHS value with RHS and returns True if both are equal. (LHS=RHS) Otherwise it returns false
!=	not equal to	Compares LHS value with RHS and returns true if both are not equal. (LHS ≠ RHS)
<	less than	
<=	less than or equals to	
>	greater than	
>=	greater than or equals to	

Logical Operator:

Operator Symbol	Name	Operation
&&	Logical AND	Checks the condition & returns true when all the conditions are true otherwise returns false.
||	Logical OR	Checks the conditions and returns false only when all the conditions are false otherwise returns true.
!	Logical NOT	It gives the nagation value of obtained result

AND

T	T	T
T	F	F
F	T	F
F	F	F

OR

T	T	T
T	F	T
F	T	T
F	F	F

Conditional or Terenary Operator:-

It is denoted by the symbol "? :". The statement ~~return~~ written question mark gets executed when the initially checked condition is true. Otherwise the statement written after ":" gets executed.

** Terenary operator can be used as alternate to if-else and else-if ladder.

Ex:

```
#include<stdio.h>
main()
{
int a,b;
Printf("enter a,b");
Scanf("%d%d",&a,&b);
a>b ? printf("a is big"): printf("b is big");
}
```

Assignment Operator:

It is denoted by the symbol =. This is used to store the values into the variable.

Ex: a=10.

Increment or Decrement Operator: Increment operator is denoted by the symbol ++ which increements the value of variable by 1.

Decrement Operator is denoted by symbol -- . Which decreases the value of a variable by 1. Each increment and decrement Operator is again classified into 2 types. They are

1, pre Increment
2, post Increment
3, pre Decrement
4, post Decrement

Pre Increment : In pre Increment the operator comes first followed by operand.

Ex: ++a

When the pre Increment operator is used along with assignment operator like

Ex: a=12; c=++a; then the value of variable a gets incremented first and then the incremented value gets assigned to c. ie after the above two steps a becomes 13 and c also becomes 13.

Post Increment : In post Increment operator the operand comes first followed by the operator.

Ex: a++.

When the post Increment operator is used along with assignment operator like

Ex: a=12;
c= a++; (c obtains value 12)

then the c value obtains 12 and a value becomes 13. the reason is the original value gets assigned first then gets incremented.

Pre Decrement: In pre Decrement the operator comes first followed by operand.

Ex: --a

When the pre Decrement operator is used along with assignment operator like

a=12;
c= --a;

then the value of variable a gets incremented first and then the incremented value gets assigned to c. that is after the above two step a becomes 11 and c becomes 11.

Post Decrement: In post Decrement operator the operand comes First followed by the operator. Ex: a--.

When the post decrement operator is used along with assignment operator like Ex: a=12; c= a--; then c value obtains 12 and a value becomes 11. The reason is the original value gets assigned first then gets decremented.

Bitwise operators:

- BitWise AND
- BitWise OR
- BitWise XOR
- Left Shift
- Right Shift

BitWise AND: It is denoted by symbol "&". It performs bitwise and operation among the bits and returns 1 if both the bits are 1 otherwise it returns 0.

Ex: 12 & 5 = ?4

```
   8 4 2 1
12 = 1 1 0 0
 5 = 0 1 0 1
-------------
   0 1 0 0 = 4
```

72 & 64 = ?64

```
    64 32 16 8 4 2 1
64 = 1  0  0 0 0 0 0
72 = 1  0  0 0 1 0 0
-------------------
     1  0  0 0 0 0 0 = 64
```

BitWise OR: It is denoted by symbol "|". It performs bitwise OR operation b/w the bits and returns 0 if both bits are 0 otherwise it returns 1.

Ex: 84 | 71 = ?87

```
    64 32 16 8 4 2 1
84 = 1  0  1 0 1 0 0
71 = 1  0  0 0 1 1 1
-------------------
87   1  0  1 0 1 1 1
```

BitWise XOR: It is denoted by symbol "^". It performs bitwise XOR operation and returns 0 if both bits are same otherwise it returns 1.

Ex: 84 ^ 71 = ?19

```
    64 32 16 8 4 2 1
84 = 1  0  1 0 1 0 0
71 = 1  0  0 0 1 1 1
-------------------
19 = 0  0  1 0 0 1 1
```

Left Shift: Left Shift operator is denoted by symbol << . The value gets double at each left shift operation i.e 10 becomes 20. 20 becomes 40 and so on...

Ex: a = 25

a << 1 = ?50

```
25 = 1 1 0 0 1
50 = 1 1 0 0 1 0
```

Right Shift: Right shift operator is denoted by symbol >> . The value gets reduce to half of its original value at each right shift operation. i.e 10 becomes 5. 5 becomes 2 ...

Ex: a = 25

a >> 2 = = ?

```
25 = 1 1 0 0 1
 6 = 0 0 1 1 0
```

8+4 = 12

```
12 = 1 1 0 0
```

4+2 = 6.

Bitwise operators:

- BitWise AND
- BitWise OR
- BitWise XOR
- Left Shift
- Right Shift

BitWise AND: It is denoted by symbol "&". It performs bitwise and operation among the bits and returns 1 if both the bits are 1 otherwise it returns 0.

Ex: 12 & 5 = ?4

	8	4	2	1
12 =	1	1	0	0
5 =	0	1	0	1
	0	1	0	0 = 4

72 & 64 = ?64

	64	32	16	8	4	2	1
64 =	1	0	0	0	0	0	0
72 =	1	0	0	1	0	0	0
	1	0	0	0	0	0	0 = 64

BitWise OR: It is denoted by symbol "|". It performs bitwise OR operation b/w the bits and returns 0 if both bits are 0 otherwise it returns 1.

Ex: 84 | 71 = ?87

	64	32	16	8	4	2	1
84 =	1	0	1	0	1	0	0
71 =	1	0	0	0	1	1	1
	1	0	1	0	1	1	1

BitWise XOR: It is denoted by symbol "^". It performs bitwise XOR operation and returns 0 if both bits are same otherwise it returns 1.

Ex: 84 ^ 71 = ?19

	64	32	16	8	4	2	1
84 =	1	0	1	0	1	0	0
71 =	1	0	0	0	1	1	1
	0	0	1	0	0	1	1

Left Shift: Left shift operator is denoted by symbol << . The value gets double at each left shift operation i.e 10 becomes 20. 20 becomes 40 and so on...

Ex: a = 25

	64	32	16	8	4	2	1
		1	1	0	0	1	

a << 1 = ?50

	64	32	16	8	4	2	1
		1	1	0	0	1	0

32 + 16 + 2 = 50

a << 2 = ?100

	64	32	16	8	4	2	1
	1	1	0	0	1	0	0

64 + 32 + 4 = 100

Right Shift: Right shift operator is denoted by symbol >> . The value gets reduce to half of its original value at each right shift operation. i.e 10 becomes 5 . 5 becomes 2 ...

Ex: a = 25

a >> 2 = = ?

	16	8	4	2	1
	1	1	0	0	1
	1	1	0	0	(1

8 + 4 = 12

	8	4	2	1
	1	1	0	0

4 + 2 = 6.

Special Operators: Special operators are of two types. They are:

1) Comma → (,)

2) Size of Operator → Size of ( )

Comma :— Comma operator is used for separating variables, statements etc.

Size of : Size of operator is used to return the size of the data type or the size of the variables declared using a particle data type. It returns the size value interms of bites.

Ex:
```
# include < stdio.h >
main ( )
{
 long int a ;
 printf ( "%d", size of (long int));
}
```

Output : 4

Write a C program to read current reading and previous reading. Using which Calculate the no. of units consumed by the customes in a Pasticular month. Generate the bill based upon the following table.

Units	Charge
0 - 100	0.50/unit
101 - 200	1/unit
201 - 300	2/unit
above 300	3/unit

```
#include < stdio.h >
main ( )
{
int cr, pr, units;
float bill;
printf (" enter values cr, pr");
scanf (" %d %d", &cr, &pr);
units = cr - pr;
if (units >= 0 && units <= 100)
{
    Bill = 0.50 * units;
}
```

```
else if (units >= 101 && units <= 200)
  {
    bill = 1 * units;
  }
else if (units >= 201 && units <= 300)
  {
    bill = 2 * units;
  }
else
  {
    bill = 3 * units;
  }
Printf("%f", bill);
}
```

Switch Statement :

Switch is a branching statement which is an alternate for else-if Laddea. Switch statement contains 4 keywords. They are

1, switch
2, case
3, break
4, default

Switch : Using switch keyword the option is checked which are to be matched with the cases.

Case : Case is the keyword we use for statements at each case one statements can be written.

break : Break is the keyword used to stop the flow of execution after matched case gets executed.

default : Default is the keyword ~~used~~ which gets executed when none of the case matched with the given option.

Syntax :

```
Switch (option)
  {
  case label 1 : statements;
                 break;
  case label 2 : statements;
                 break;
  default : statements;
  }
```

Write a c program to read an option from the keyboard and display the city names as per the options.

option 1 - Vijayawada

option 2 - Hyderabad

option 3 - Vizag

option 4 - chennai

Display not a correct option if other option is entered.

```
#include < stdio.h>
main( )
{
int a;
Printf(" enter a value");
Scanf("%d", &a);
Switch(a)
{
case 1 : printf(" vijayawada");
        break;
Case 2 : printf(" Hyderabad");
        break;
Case 3 : printf("vizag");
        break;
Case 4: printf("chennai");
        break;
default: printf(" not a correct option");
 }
}
```

Write a C program to perform arithmetic operation based on the option enter.

1- addition

2- Subtraction

3- Multiplication

4- Division

5 - Modulo

```
# include < stdio.h>
main( )
{
```

```
int c, a, b, n;
printf("enter a, b values");
scanf("%d %d", &a, &b);
printf("enter ur option");
scanf("%d", &n);
switch (n)
{
case 1 : c = a+b ;
         printf("%d", &c);
         break;
case 2 : c = a-b;
         printf("%d", &c);
         break;
case 3 : c = a * b;
         printf("%d", &c);
         break;
case 4 : c = a/b ;
         printf("%d", &c);
         break;
case 5 : c = a%b;
         printf("%d", &c);
         break;
default : printf("not a correct option");
  }
}
```

```
char ch[10];
scanf("%s", ch);
printf("%s", ch);
}
```

```
char ch;
scanf("%c", &ch);
printf("%c", ch);
}
```

Write a C program to read id, name, gender, marks of a student and display those details.

```
#include <stdio.h>
main()
{
int id;
char name[20];
char gender;
float marks;
printf("enter ur id");
scanf("%d", &id);
fflush(stdin);
printf("enter ur name");
scanf("%s", name);
fflush(stdin);
printf("enter ur gender");
scanf("%c", gender);
```

char name[20]; → character array (used for strings) → group of characters. Ex: names, i.e RISHITHA

char gender; → data type used for single character.

this step must be used before character & string readings.

'&' is not used

format specifier for strings

```
Printf("enter uv marks");
Scanf("%f", &marks);
printf("%d %s %c %f", id, name, gender, marks);
}
```

Predict the Output of the following programme.

1)
```
#include <stdio.h>
main()
{
int x=25, a=0, b=3;
if (x>=25)
{
  int c=a+b;
 Printf("%d", c);
}
else if(1)
{
   Printf("%d", a-b);
}
else
{
  Printf("hello");
}
}
```

Output:
5

2)
```
#include <stdio.h>
main()
{
if(1)
printf("hello");
else
printf("bye");
}
```

Output:
hello
1 indicates true & 0 indicates false.
Other than 1 also we will indicate as true.

```
#include <stdio.h>
main()
{
  int i=3;
  i+=5; ⇒ i=i+5;
printf("%d", i);
}
```

Looping Statements: The other name for looping statement is Iterative statements. There are three types of loops.

1, for loop

2, While loop

3, do-While loop

for loop :- for loop is catagorised into 4 steps.

1. initialization
2. Condition checking
3. Statements
4. Increment/decrement.

Syntax :- for (initialization ; Condition checking ; increment/decrement)

```
{
 statements;
}
```

The control starts with initialization part followed by condition checking. If the condition is true the control enters the block of for loop executes its statements and then the control goes to increment or decrement part followed by again condition check & the process goes on until the condition fails.

Write a c program to display the message Hello for 100 times using for loop.

```
#include <stdio.h>
main()
{
  int i;
  for (i=1; i<=100; i++)
  {
  printf("hello \n");
  }
}
```

Write a c program to display the numbers between 1 to 100.

```
#include <stdio.h>
main()
{
```

```
int i;
for (i=1; i<=100; i++)
{
  printf("%d\n", i);
}
}
```

Write a cprogram to display the numbers b/w 1000 to 500.

```
#include <stdio.h>
main()
{
  int i;
for (i=1000; i>=500; i--)
{
  printf("%d\n", i);
}
}
```

Write a cprogram for sum of n natural numbers using for loop.

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

Write a C program to calculate the sum of even numbers b/w 1 to n.

```
#include <stdio.h>
main()
{
 int i, x=0, n;
printf("enter n");
```

```
scanf("%d", &n);
for (i=1; i<=n; i++)
1<=6;
{
 if (i%2==0)
   x = x+i;
}
Printf("%d", x);
}
```

* Write a c program for sum of even numbers and sum of odd numbers separetly between 1 to n.

```
#include<stdio.h>
main()
{
scanf("%d", &n);
int i, x=0, y=0, n;
for (i=1; i<=n; i++)
1<=9
{
if (i%2==0)
  {
   x = x+i;
  }
else
  {
   y = y+i;
  }
}
Printf("even = %d \n odd = %d", x, y);
```

Write a c program to print the factors of n.

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

```
{
  Printf("%d\t", i);
  }
}
```

Write a C program to find the sum of factors of a given number.

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

Write a C program to read the numbers and check whether it is perfect number or not. 1+2+3=6

```
#include<stdio.h>
main()
{
int n,i,x=0;
Printf("enter n");
Scanf("%d",&n);
for(i=1; i<n;i++)
{ if(n%i==0)
    {
     x=x+i;
    }
}
Printf("%d",x);
if(i==n)
printf("perfect number");
}
```

```
else
}
Printf("not a perfect number");
}
}
```

Write a C program to count no. of factors of a given number.

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

Write a C program to check the number is prime number or not.
(if no. of factors are 2 then it is a prime number)

```
#include<stdio.h>
main()
{
int n,i,x=0;
Printf("enter n");
scanf("%d",&n);
for(i=1; i<=n; i++)
{ if(n%i==0)
    {
     x++;
    }
}
if(x==2)
Printf("prime number");
else
Printf("not a prime number");
}
```

While loop: While loop is an iterative statement which uses the keyword While. While loops also contains 4 major steps just as for loop.

They are

1, Initialization

2, Condition checking

3, Statements

4, Increment / Decrement

Syntax:

```
initialization;
While (condition checking)
{
statements;
increment/decrement;
}
```

Write a C program to print (read) the numbers b/w 1 to n using while loop.

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

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

Write a C program to display the numbers b/w 10 to 1a using while loop.

```
#include<stdio.h>
main()
{
int i;
i=10;
while(i>=1)
{
```

```
Printf("%d",i);
    i--;      ;++ => 11 to 20.
 }
}
```

Write a C program to calculate the sum of even numbers b/w 1 to n.

```
#include<stdio.h>
main()
{
int i,n, x=0;
Printf("enter n");
scanf("%d", &n);
i=1;
while(i<=n)
{
if (i%2==0)      if(i%2==0)
                   x=x+i;
   x=x+i;    =>   i++;
   i++;          }
}
Printf("%d", x);
}
```

Write a C program to print multiplication table

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

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

What is the output for the following program.

```
main()
{
int i;
for(i=100; i>=1; i=i/2)
{
 Printf("%d",i);
 }
}
```

100
50
25
12
6
3
1

Write a C program to read a number and display individual digits of a number?

```
#include<stdio.h>
main()
{
int n,r;
Printf("enter n");
Scanf("%d",&n);
While (n>0)
{
r=n%10;
Printf("%d\t",r);
n=n/10;
}
}
```

Write a C program to read a number and display square of individual digits of a number?

```
#include<stdio.h>
main()
{
int n,r;
Printf("enter n");
Scanf("%d",&n);
While (n>0)
{
r=n%10;
printf("%d\t",r*r);
n=n/10;
}
}
```

Write a C program to read a numbes and calculate sum of digits of a number.

```
#include<stdio.h>
main()
{
int n,r,X=0;
Printf("enter n");
```

```
scanf ("%d", &n);
while (n>0)
{
r = n%10;
x = x+r;
n = n/10;
}
printf ("%d", x); → to know final output only.
}
```

```
{
r = n%10;
x = x+r;
printf ("%d", x);
n = n/10;
}
```

→ every time it will print the updated loop.

Write a C program to read a number and calculate the sum of even digits and odd digits seperately and display the results.

```
#include <stdio.h>
main() {
int n, r, x=0, y=0;
printf ("enter n");
scanf ("%d", &n);
while (n>0)
{
r = n%10;
if (r%2 == 0)
x = x+r;
else
y = y+r;
n = n/10;
}
printf ("%d", x);
}
```

if we want to get count

```
if (r%2 == 0)
x++;
else
y++;
```

Write a C program to read a number and check whether it is angstorm number or not.

angstrom number: Sum of cubes of individual digits of a number is equal to the same number

Ex: 153

$3^3 + 5^3 + 1^3 \Rightarrow 153$.

```
#include <stdio.h>
main() {
int n, r, x=0, z;
printf ("enter n");
scanf ("%d", &n);
```

```
z = n;
while (n>0)
{
  a = n%10;
  x = x+a*a*a;
  n = n/10;
}
if (z==x)
printf("amstrong");
else
printf("not an amstrong");
}
```

Nested - for loops:

for loop within another for loop is said to be the nested for loop. Here when we come to the block of outer for loop at after verifying the condition; if again the for loop is available in the place of statement block then the behaviour of inner for loop doesn't change and executes in the same order i.e insitalization, Condition checking, Statements, increment/decrement. and the inner for loop runs repeatedly until the condition gets failed then only the control get back to the increment/decrement section of outer for loop.

Outer for loop deals with rows and inner for loop deals with column.

Write a C-program to print the following pattern?

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

```
# include < stdio.h >
main ()
 {
  int i,j;
for (i=1; i<=5; i++)
{
 for (j=1; j<=5; j++)
   {
    printf("*  ");
   }
printf ("\n");
}
}
```

Write a c program to print the following pattern?

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

```
#include<stdio.h>
main()
{
int i,j;
for(i=1;i<=5;i++)
{
for(j=1;j<=5;j++)
  {
  printf("%d",i);
  }
printf("\n");
  }
}
```

Write a c program to print the following pattern?

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

```
#include<stdio.h>
main()
{
int i,j;
for(i=1;i<=5;i++)
{
for(j=1;j<=5;j++)
 {
 printf("%d",j);
  }
printf("\n");
  }
}
```

Write a c program to print the following pattern? (increasing pattern)

inner for loop condition is based on outer for loop.

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

```
#include<stdio.h>
main()
{
int i,j;
for(i=1;i<=5;i++)
{
for(j=1;j<=i;j++)
{
  printf("%d",i);
 }
printf("\n");
}
```

Write a c program to print the following pattern.

```
#include<stdio.h>
main()
{
int i,j;
for(i=5; i>=1;i--)
{
  for(j=1; j<=i; j++)
  {
  printf("%d",j);
 }
 printf("\n");
}
}
```

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

Write a c program to print the following pattern:

```
#include<stdio.h>
main()
{
int i,j;
for(i=1; i<=5; i++)
{
 for(j=1; j<=i; j++)
 {
   printf("%d",j);
 }
printf("\n");
 }
}
```

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

Write a c program to print the following pattern:

```
#include<stdio.h>
main()
{
int i,j;
for(i=5 ; i>=1;i--)
{
 for(j=5; j<=i; j++)
```

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

```
}
printf("%d", i);
}
printf("\n");
}
}
```

Write a c program to print the pattern.

```
#include<stdio.h>
main()
{
for(i=1; i<=5; j++)
{
for(j=1; j<=5; j++)
{
x=x+1;
printf("%d", x);
}
Printf("\n");
}
}
```

1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25

Write a c program for swapping of two numbers.

```
#include<stdio.h>
main()
{
int a,b,c;
printf("enter a,b values");
scanf("%d %d", &a, &b);
c=a;
a=b;
b=c;
printf("%d %d", a,b);
```

output:
a=12 : b=45
45 12

Pointers: pointers are used to store the address of a variable.
Integer pointers can store the address of integer variables
Character pointers can store the address of character variables and so on.

Ex:
```
#include<stdio.h>
main()
{
int a=76, *p;
```

→ Storing the address of a variable 'a' into the pointer p

```
p = &a;          → it stores big value.
printf("%u\n", p); → using pointer p, displaying the address of 'a'.
printf("%d", *p); → using pointer p, displaying the value of variable 'a'.
}
```

Output:

6487572 → address of a

76 → value of a

Find the output of following program.

```
#include <stdio.h>
main()
{
int a=76, b=22, c=12, *p, *q, *r, x, y;
p=&a;
r=&c;
q=&b;
x=*p;        76
y=*r+*q+x;   12+22+76 = 110
y+=x;        y=y+x = 110+76 = 186
printf("%d\n", y);   186
printf("%d", *p);    76
}
```

if we are changing the value after initialization then after the changing value only we consider.

Ex:

```
int a=76, b=45, *p, *q;
p=&a;
q=&b;
a=123;
*q=176;
printf("%d\n", *p);
printf("%d", b);
```

output: 123, 176

Find the output of the following program.

```
#include <stdio.h>
main()
{
 int a=76, *p;
 p=&a;
 *p = *p*4;   76×4 = 304
printf("%d\n", a);
}
```

Find the output of the following program.

```
#include <stdio.h>
main()
 {
int a=76, b=22, c=12, *p, *q;
 p=&a;
```

```
q = &c;
b = b + *p + *q;   220 + 12 + 110
*p = b * 2;   110 * 2 = 220
printf("%d\n", *p + *q + a);   220 + 12 + 220 = 452
```

DO - While:

Find the output of the following program.

```
#include<stdio.h>
main()
{
printf("enter n");   5
scanf("%d", &n);
i=1;
do
{
x=x+5;   → x=0+5 | 5+5=10 | 10+5=15 | 15+5=20 | 20+5=25
printf("%d", x);   5  10  15  20  25
    i++;   i=2  i=3  i=4  i=5  i=6
} while (i<=n);   2<=5  3<=5  4<=5  5<=5  6<=5 (so it will not go back)
}
```

Write a program to find the output of the program.

```
#include<stdio.h>
main()
{
int i, n, x=0, y;
Printf("enter n");
scanf("%d", &n);
 i=1;
 do
{
x=x+5;   x=5  x=10  x=15  x=20  x25
y=x%2;   1  0  1  0  1
printf("%d", y);   1  0  1  0
  i++;   i=2  i=3  i=4  i=5
} while (i<=n);
}
```

In do while there are two keywords are being used. They are do, while. It is also a looping statement which executes repeteadely

until the condition fails but there are few differences b/w while and do while

While	do-While
* This is entry condition loop	* This is exist condition loop.
* If the condition fails for the 1st time itself then none of the statements in the while loop gets executed even one time also.	* The statement gets executed atleast once in do while loop even though condition fails for the 1st time.

* Syntax:
while (condition)
{
statements;
increment / decrement;
}
	* Syntax:
	do
	{
	statements;
	increment / decrement;
	} while (condition);

Functions:

Create a function klu which reads input from keyboard and check whether the element is even or odd.

```
#include<stdio.h>
klu
{
  int n;
printf("enter n");
scanf("%d",&n);
if(n%2==0)
Printf("even");
else
Printf("odd");
}
main()
{
 klu();
}
```

with parameter with return type
with parameter without return type
without parameter with return types
without parameter without return type

Output:

```
enter n : 45
  odd
```

Create a function phone which takes 2 input values from calling function through parameters and displays the biggest of 2 numbers.

```
#include<stdio.h>
phone (int a, int b);
{
 if (a>b)
printf("enter a,b values");
scanf("%d %d", &a, &b);
Printf("a is big");
else
Printf("b is big");
}
main()
{
 int a,b;
Print F("enter a,b values");
scanf("%d %d", &a, &b);
Phone (a,b);
}
```

wrong

Write a C program to create a function phone which reads the n value from the keyboard and calculate the sum of n natural numbers.

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

Precedence and Associativity of operators

In C language operators are assigned with some ranks. The other name for rank of an operator is precedence.

While evaulating the expression highest rank operators are evauted first.

While evaulating the expression, When two or more operators of same rank (precedence) are to be evaluated then associativity of operators will decide wheather they are to be evaulated from left to right (or) right to left.

	Precedence	Associativity
1)	( ), ++(post), --(post)	⟶ right to left
2)	++(pre), --(pre)	⟶ Left to right
3)	*, /, %	⟶ L to R
4)	+, -	⟶ L to R
5)	<<, >>	⟶ L to R
6)	<, <=, >, >=	⟶ L to R
7)	&	⟶ L to R
8)	^	⟶ L to R
9)	|	⟶ L to R
10)	&&	⟶ L to R
11)	||	⟶ L to R

1) 12/7 * 3/4 - 3

⟶ 1 * 3/4 - 3

⟶ 3/4 - 3

⟶ 0 - 3

⟶ = -3

2) (12-3) / (8*4) * 5

⟶ 9 / (8*4) * 5

⟶ 9/32 * 5

⟶ 0 * 5

⟶ 0

Files: File is a collection of records(data). Files are used to store the data permenantly.

Ex: doc.txt

text file

Functions in a file:

fopen(): fopen is the function used to open the file.

fclose(): fclose is the function used to close the file.

fprintf(): fprintf is the function used to store the data into the file.

fscanf(): fscanf is the function used to get the data from the file.

Filepointers: File pointer is used to connect with the file and the file pointer must be created using the name FILE.

Ex: FILE *p;

Write a c program to create a file board.txt. Read two integer numbers and calculate the sum of numbers then store the result into the file.

```
#include<stdio.h>
main()
{
int a,b,c;
FILE *p;
p=fopen("board.txt","w");
Printf("enter a,b values");
scanf("%d%d",&a,&b);
c=a+b;
fprintf(p,"%d",c);
fclose(p);
}
```

file name ↑ , "w" → Write mode

Write + read mode ↑

Write a c program to open a file phone.txt in 'w' mode. Read two integer numbers from the user & calculate addition of two numbers and store the result into the file. Retrive the stored value from the same file and then update the result then display the result to the user.

```
#include<stdio.h>
main()
{
int a,b,c,z=123;
```

```
FILE *p;
P = fopen("phone.txt", "w+");
Printf("enter a,b values");
scanf("%d %d", &a, &b);
c = a+b;
fprintf(P, "%d", c);
fscanf(P, "%d", &c); → reading the value from the file
z = z+c;
Printf("%d", z);
fclose(p);
}
```

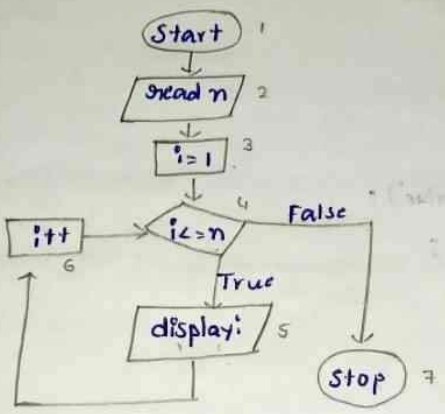
Write a C program to create a file laptop.txt in 'w+' mode. Read the values from the user and then calculate addition of two numbers then store the result into the same file. Create one more file charger.txt in write mode. Get the value from the 1st file and copy into the second file.

```
#include<stdio.h>
main()
{
int a,b,c;
FILE *P,*q;
P = fopen("laptop.txt", "w+");
Printf("enter a,b values");
scanf("%d%d", &a, &b);
c = a+b;
fprintf(P, "%d", c);
fscanf(P, "%d", &c);
q = fopen("charger.txt", "w");
fprintf(q, "%d", c);
fclose(p);
fclose(q);
}
```

Flow Chart for Loops:

Write a c program to display the numbers 1 to n using for loop.

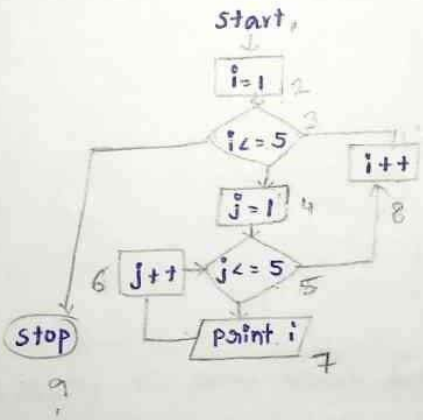
```
#include<stdio.h>
main()
{
int i,n;
for(i=1;i<=n;i++)
{
printf("%d",i);
}
}
```



Algorithm

1, Start
2, read n value
3, initialize i=1
4, check the condition i<=n
if true goto step 5
otherwise goto step 7
5, display i
6, i++, goto step 4
7, Stop

```
#include<stdio.h>
main()
{
int i,n
for(i=1;i<=5;i++)
{
for(j=1;j<=5;j++)
{
printf("%d",i);
}
}
}
```



Reversing of a number:

Write a c program to read the number n and display the reverse of a number: Ex: 123 → 321

```
#include<stdio.h>
main()
{
int r, x=0, n;
Printf("enter n value");
Scanf("%d", &n);
While(n>0)
{
 r=n%10;
 x = x*10+r;
 n= n/10;
}
Printf("%d", x);
}
```

Write a c program to read a number and check whether it is palindrome or not.

Palindrome : Reverse of a number = Same number.

Ex: 121, 343 ...

```
#include<stdio.h>
main()
{
int r, x=0, n, z;
printf("enter n value");
scanf("%d", &n);
n=z;
While (n>0)
{
 r= n%10;
 x = x*10+r;
 n= n/10;
if (x==z)
printf("palindrome");
else
printf("not a palindrome");
}
}
```

fibonacci series :

Write a c program for fibonacci series.

Start with 0 1

```
#include< stdio.h >
main()
{
int a=0, b=1, c, i;
Printf(" %d %d", a, b);
for(i=1; i<=10; i++)
{
  c=a+b;
Printf(" %d", c);
a=b;
b=c;
}
}
```

Write a cprogram for Lucas prog series.

Start with 2 1

```
#include< stdio.h >
main()
{
int a=2, b=1, c, i;
Printf(" %d %d", a, b);
for(i=1; i<=10; i++)
{
  c=a+b;
Printf(" %d", c);
a=b;
b=c;
}
}
```

Write a cprogram to display the prime numbers between 1 to n.

```
#include< stdio.h >
main()
{
int n, i, x=0, j;
Printf("enter n");
Scanf(" %d", &n);
```

limit

```
for(i=1; i<=n; i++)
{
  x=0;
  for(j=1; j<=i; j++)
  {
    if(i%j==0)        j is a factor for i
    x++;
  }
  if(x==2)
  Printf("%d", i);
  }
}
```

i = 1, 2 - - - - -
the input given for checking whether it is prime or not.

Global Variable: A variable which is declared outside the function is said to be a global variable and that variable can be accessed by any function.

Example:

```
#include<stdio.h>
int a=12;
main()
{
Printf("%d", a);
light();
}
light()
{
  Printf("%d", a);
}
```

In the above example, a is visible to both the functions as it is a global variable. (Variable)

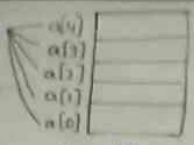
Arrays: An array is a collection of elements of similar data type.

Syntax: data type array name[size];

Ex: int a[5];

* In the above example an array with name a with size 5 of integer type is declared.
* It means we have created five memory slots of integer type.

Ex: int a[5];



* In the above example the 5 which is written during declaration only represents the size except that wherever the number used in [ ] represents the index number or slot number

Assigning values to arrays:-

Assignment of values to arrays can be done in two ways.

1) Assignment during declaration

Ex: int a[5] = {12, 34, 76, 89, 95}; ⇒

a[4]	95
a[3]	89
a[2]	76
a[1]	34
a[0]	12

2) Assignment using scanf along with for loop

```
Ex: int a[5], i;
    for (i=0; i<5; i++)
    {
     scanf ("%d", &a[i]);
    }
```

Predict the output of following program:

```
#include<stdio.h>
main()
{
int a[5] = {45,67,23,78,98};
Printf("%d\n", a[3]);
Printf["%d\n", a[0]+a[4]);
}
```

Output:

78

143

Write a cprogram to read elements into an array and display those elements.

```
#include<stdio.h>
main()
{
int a[5], i;
Printf("enter array elements");
for(i=0; i<5; i++)
{
scanf("%d", &a[i]);
```

```
}
Printf ("elements are \n");
for (i=0; i<5; i++)
{
Printf ("%d", a[i]);
}
}
```

Write a C program to read elements into an array & calculate sum of array elements & then display the result.

```
#include<stdio.h>
main()
{
int a[5], i, x=0;
Printf ("enter array elements");
for (i=0; i<5; i++)
{
 scanf ("%d", &a[i]);
}
Printf (" result is \n");
for (i=0; i<5; i++)
{
 x=x+a[i];
}
Printf ("%d", x);
}
```

for only even sum → `if (a[i]%2==0)`

for both even & odd

```
if (a[i]%2==0)
x=x+a[i];
else
y=y+a[i];
Printf
```

Storage classes : There are 4 storage classes

keywords

1) Automatic ⟶ auto — garbage
2) Registered ⟶ register
3) Static ⟶ Static — default value is 0.
4) External ⟶ extern

Find the output of the following program

```
#include <stdio.h>
main()
{
auto int a=12;
 {
  auto int a=45;
  Printf ("%d \n", a);
```

```
}
Printf("%d", a);
}
```

```
auto int a=12;
a=45;
Printf("%d\n", a);
Printf("%d", a);
45
45
```

Output:
45
12

Static Keyword: Static keyword default value is 0. The variable which is declared with the static remains active between the function calls.

Find the output of following program. (automatic)

```
#include<stdio.h>
main()
{
watch();
watch();
watch();
}
watch()
{
 int a=12, b=13;
  a++;
  printf("%d \n", a);
}
```

Output: 13
13
13

Find the output of following program. (static)

```
#include<stdio.h>
main()
{
watch();
watch();
watch();
}
watch()
{
Static int a=12;
      a++; → (13) + 1 = (14) + 15
  Printf("%d\n", a);
}
```

output: 13
14
15

Static variables remains active between the function calls.

Find the output of following program.

```
#include<stdio.h>
main()
{
 watch();
watch();
watch();
}
watch()
{
Static int a;
a = a+12;
Printf("%d\n", a);
}
```

output: 12
24
36

0+12 = 12
12+12 = 24
24+12 = 36

Recurssion: → calling a function within itself.

```
#include<stdio.h>
main()
 {
int n, f;
Printf("enter the value of n");
scanf("%d", &n);
} f = fact(n);
   Printf("%d", f);
int fact(int n) return o;
 {
 if (n==0 || n==1)
return 1;
else
 return * fact(n-1);
}
```

Write a c program to display the natural numbers between 1 to n using recussion.

```
#include<stdio.h>
main()
{
int n, j = 1;
Printf("enter n");
Scanf("%d", &n);
Sunday(i,n);
}
```

```
Sunday(int i, int n)
{
 if(i<=n)
 {
  Printf("%d\n",i);
  i++;
Sunday(i,n);
}
```

Write a c program to display from n to 1 using recurssion.

```
#include<stdio.h>
Sunday(int i, int n);
main()
{
int n, i=1;
Printf("enter n");
Scanf("%d",&n);
Sunday(i,n);
}
Sunday(int i, int n)
{
if(n>=i)
{
Printf("%d\n",n);
n--;
Sunday(i,n);
}
}
```

Find the output of the following

```
#include<stdio.h>
#include<math.h>
main()
{
int x=16,y,z;
y=Sqrt(x);
z=pow(x,2);
Printf("%d\n%d", y,z);
}
```

output:

4

256

Find the output of the following.

```
#include <stdio.h>
main()
{
int n;
Printf("enter n");
scanf("%d", &n);
Sunday(n);
}
Sunday(int n)
{
 Static int a;
a++;
if(a<=n)
{
 Printf("%d",a);
Sunday(n);
  }
}
```

output
1 2 3 4 5

Stack memory:
2 * fact(1) =) 2 x 1 = 2
3 * fact(2) =) 3 x 2 = 6
4 * fact(3) =) 4 x 6 = 24
5 * fact(4) =) 5 * 24 = 120

GCD

```
#include<stdio.h>
main()
{
int divisor, dividend, z;
Printf("enter dividend, divisor");
scanf("%d %d", &dividend, &divisor);
z= gcd(dividend, divisor);
Printf("%d", z);
}
gcd(int dividend, int divisor)
{
 int remainder;
remainder = dividend % divisor;
if(remainder == 0)
return divisor;
else
gcd(divisor, remainder);
}
```

fibonacci Series

```
#include<stdio.h>
main()
{
int a=0,b=1,n;
Printf("enter n");
scanf("%d",&n);
Printf("%d %d",a,b);
fib(a,b,n);
}
fib(int a,int b,int n)
{
 int c
 if(n>0)
  {
  c=a+b;
  Printf("%d",c);
   a=b;
   b=c;
fib(a,b,n-1);
 }
}
```

Two-dimensional array:

Two-dimensional array is having Two index points one represent rows another represents columns.

Syntax for declaration of 2D-Array:

arrayname [rowsize] [column size]

Ex: a[2][2] $\begin{bmatrix} 00 & 01 \\ 10 & 11 \end{bmatrix}$

Write a C program to read elements into 3×3 matrix and display those elements.

```
#include<stdio.h>
 main()
{
int i,j,a[3][3];
Printf("enter array elements");
```

```
for(i=0; i<3; i++)
{
for(j=0; j<3; j++)
{
scanf("%d", &a[i][j]);
  }
}
for(i=0; i<3; i++)
{
for(j=0; j<3; j++)
{
Printf("%d", a[i][j]);
}
Printf("\n");
  }
}
```

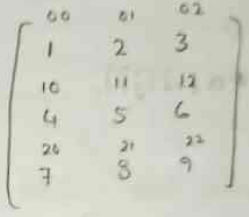
Write a c program to read elements into 3×3 matrix and display the transpose of a matrix.

```
#include<Stdio.h>
main()
{
int i,j, a[3][3];
Printf("enter array elements");
for(i=0; i<3; i++)
{
for(j=0; j<3; j++)
{
scanf("%d", &a[i][j]);
  }
}
for(i=0; i<3; i++)
{
for(j=0; j<3; j++)
{
Printf("%d", a[j][i]);
}
```

```
Printf("\n");
  }
}
```

Write a c program to read the elements in 3x3 matrix and display the diagonal elements.

```
#include<stdio.h>
main()
{
int i,j,a[3][3];
Printf("enter array elements");
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
Scanf("%d",&a[i][j]);
  }
}
for(i=0;i<3;i++){
for(j=0;j<3;j++)
{
 if(i==j)
{
Printf("%d\n",a[i][j]);
 }
 }
Printf("\n");
}
}
```



1+5+9 = 15

Output: 15.

Write a C program to find the trace of a matrix.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

$1+5+9 = 15$

```
#include<stdio.h>
main()
{
int a[3][3],i,j,x=0;
printf("enter array elements");
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
scanf("%d",&a[i][j]);
}
}
printf("elements are\n");
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
if(i==j)
x=x+a[i][j];
}
}
printf("%d",x);
}
```

Write a C program to read elements into the nxn matrix and display all the diagonal elements (both principal & opp diagonal elements).


```
#include<stdio.h>
main()
{
int a[10][10],i,j,n;
printf("enter n");
scanf("%d",&n);
printf("enter array elements");
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
scanf("%d",&a[i][j]);
```

```
}
}
Printf("elements are \n");
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
if((i==j)||(i+j==n-1))
Printf("%d",a[i][j]);
}
}
```

Write a c program to read elements into nxn matrix and calculate the sum of opposite diagonal elements.

```
#include<stdio.h>
main()
{
int a[10][10],i,j,n;
Printf("enter n");
Scanf("%d",&n);
Printf("enter array elements");
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
Scanf("%d",&a[i][j]);
}
}
Printf("elements are \n");
for(i=0;i<n;i++)
{
for(j=0;j<n:j++)
{
if(i+j==n-1)
Printf("%d",a[i][j]);
}
}
```

Write a c program to read elements into one-dimensional array (1d array) and then double the even elements and replace the odd elements with -1.

```
#include<stdio.h>
main()
{
int a[10],n,i;
Printf("enter array elements");


for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
for(i=0;i<n;i++)
{
 if (a[i]%2==0)
 a[i] = a[i]*2;
 else
a[i] = -1;
}
Printf("updated array is\n");
for(i=0;i<n;i++)
 {
 Printf("%d", a[i]);
 }
}
```

Write a C program to read elements into 3x3 matrix and display row wise sum.

```
#include<stdio.h>
main()
{
int a[10],n,i;
Printf("enter array elements");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
for(i=0;i<3;i++)
{
  x=0;
for(j=0;j<3;j++)
{
x=x+a[i][j];
Printf("%d",x);
}
```

Write a C program to read elements into nxn matrix and calculate the sum of principle diagonal elements and sum of secondary diagonal elements seperately. Display the result of their difference.

```
#include<stdio.h>
main()
{
int a[10][10], i, j, n, x=0, y=0;
Printf("enter n");
scanf("%d", &n);
Printf("enter array elements");
for(i=0; i<n; i++)
{
for(j=0; j<n; j++)
{
scanf("%d", &a[i][j]);
}
}
for(i=0; i<n; i++)
{
for(j=0; j<n; j++)
{
  if(i==j)
    x= x+a[i][j];
  if(i+j==n-1)
  y=y+a[i][j];
}
}
Printf("result = %d", x-y);
}
```

Write a c program to read elements into two nxn matrices. calculate the addition of two matrices.

```
#include<stdio.h>
main()
{
int a[10][10], b[10][10], c[10][10], i, j, n;
Printf("enter n");
scanf("%d", &n);
Printf("enter 1st matrix elements\n");
```

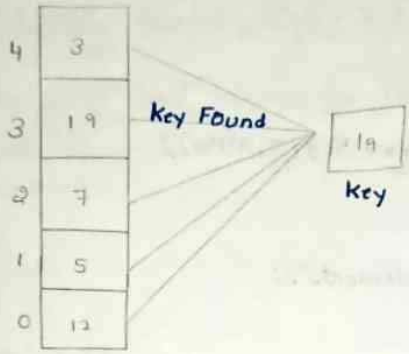
```
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
scanf("%d",&a[i][j]);
}
}
Printf("enter 2nd matrix elements\n");
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
scanf("%d",&b[i][j]);
}
}
for(i=0;i<n;i++)
{
for(j=0;j<n;i++)
{
c=[i][j]=a[i][j]+b[i][j];
}
}
Printf("matrix addition is\n");
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
Printf("%d",c[i][j]);
}
Printf("\n");
}
}
```

Searching and Sorting:

* Searching Technique: The process of searching an element is known searching technique. There are two types of searching techniques are there. They are

1) Linear Search Technique
2) Binary Search Technique

1) Linear Search Technique: The process of searching element linearly is known as linear search technique.



Write a C program to read the elements into an array and check whether the key element is available or not using linear search technique.

```
#include<stdio.h>
main()
{
int x=0, i, key, n;
Printf("enter n");
scanf("%d", &n);
 int a[n];
 Printf("enter array elements");
for(i=0; i<n ; i++)
{
scanf("%d", &a[i]);
}
Printf("enter key");
Scanf("%d", &key);
for(i=0; i<n; i++)
{
 if(a[i]==key)
    {
     x++;
  break;
 }
}
```

```
if (x==0)
Printf("not found");
else
Printf("found");
}
```

Binary Search Technique:

Write a C program to read the elements into an array in ascending order and search for the key element using binary search technique.

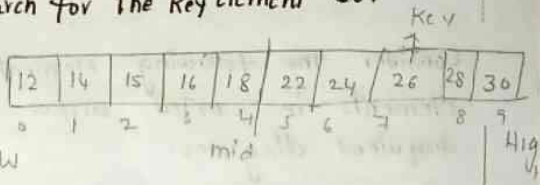
```
#include<stdio.h>
main()
{
int low, high, mid, i, x=0, key, n, a[100];
Printf("enter n");
scanf("%d", &n);
Printf("enter array elements");
for(i=0; i<n; i++)
{
 scanf("%d", &a[i]);
}
Printf("enter key");
scanf("%d", &key);
low = 0;
high = n-1;
while(low<=high)
{
 mid = (low+high)/2;
if (key > a[mid])
{
  low = mid+1;
}
else if ( key < a[mid])
{
 high = mid-1;
}
else
  {
   x++;
  break;
  }
}
```

```
if (x == 0)
printf("not found");
else
printf("found");
}
```

Unlike linear Search Technique Where we look for key element linearly one by one, In binary Search Technique We reduce no. of searches by dividing the entire array into two parts by using the formula

$mid = \frac{(Low + High)}{2}$. Initially low = 0 ; high = n-1

Consider the following array and Search for the key element 26.



initially Low = 0

High = 9

$mid = \frac{0+9}{2}$

$= 4$

a[mid] = a[4] = 18

Key = 26

As key > 18

Low = mid + 1

= 4 + 1

= 5

Now = mid = (low + high) /2

= (5 + 9)/2

= 14/2

= 7

a[mid] ⇒ a[4] = 18

Age ⟹ 26

as key > 18

mid = 7

a[mid] ⇒ a[7] = 26

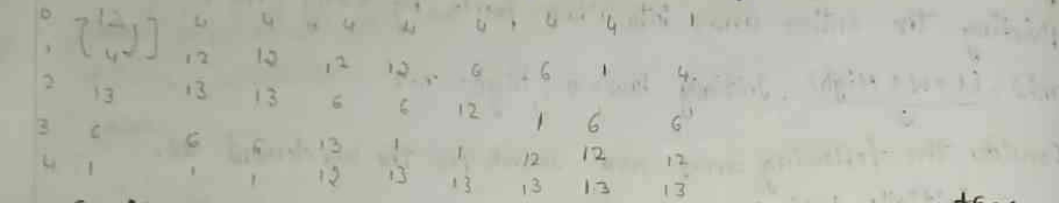
key = a[mid]

Here element found

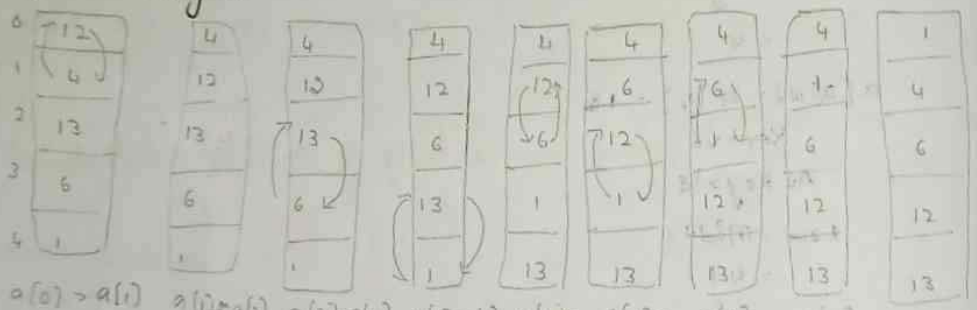
The process of arrangement of elements either in ascending order or decending order is set to be a sorting. That technique used for sorting is said to sorting Technique.

Ex: Bubble sort.

Bubble sort: Bubble sort is a sorting technique which compares two elements at a time & perform sorting as per requirement.



Consider the following elements 12, 4, 13, 6, 1 and arrange those elements in sorted arrow using bubble sort technique with required diagrams.



write a c program to read n elements into array. Arrange them into sorting order using bubble sort technique.

```
#include <stdio.h>
main()
{
Int a[100], i, n, c, j;
Printf("enter n");
Scanf("%d", &n);
Printf("enter array elements");
for(i=0; i<n; i++)
{
scanf("%d", &a[i]);
}
for (i=0; i<n; i++)
{
for (j=0; j<n-i-1; j++)
{
  if (a[j] > a[j+1])
{
c = a[j];
```

```
    a[j] = a[j+1];
    a[j+1] = c;
    }
  }
}
printf("sorted elements\n");
for(i=0; i<n; i++)
{
  printf("%d\t", a[i]);
  }
}
```

Write a C program for multiplication of a matrices.

```
#include<stdio.h>
main()
{
 int a[10][10], b[10][10], c[10][10], i, j, n;
 printf("enter n");
 scanf("%d", &n);
 printf("enter 1st matrix elements");
 for(i=0; i<n; i++)
 {
 for(j=0; j<n; j++)
 {
 scanf("%d", &a[i][j]);
 }
 }
 printf("enter 2nd matrix elements");
 for(i=0; i<n; i++)
 {
 for(j=0; j<n; j++)
 {
 scanf("%d", &b[i][j]);
 }
 }
 for(i=0; i<n; i++)
 {
 for(j=0; j<n; j++)
    c[i][j]=0;
```

```
for(k=0; k<n; k++)
{
c[i][j] = c[i][j] + a[i][k] * b[k][j];
}
}
}
Printf("matrix multiplication is \n");
for(i=0; i<n; i++)
{
for(j=0; j<n; j++)
{
Printf("%d", c[i][j]);
}
Printf("\n");
}
```

Strings: A String is a group of characters.

#include<string.h> is the headerfile used for strings.

'\0' → is the indication of end of the string.

String Handling Functions (or) String Manupulation functions.

Some of the string handling function are

- Strlen()
- Strrev()
- Strupr()
- Strlwr()
- Strcat()
- Strcmp()
- Strcmpi()
- Strcpy()
- Strncat()
- Strncmp()
- Strncpy()

1) Strlen() : is the function used to find the length of the string. i.e no. of characters in a string.

Ex:
```
#include<string.h>
main()
{
```

```
char a[20];
int z;
printf("enter string\n");
gets(a);
z = strlen(a);
printf("%d", z);
}
```

Input : Rishitha

output: 8

strrev() : is the function used to reverse the string.

Ex:
```
#include<stdio.h>
main()
{
char a[20];
printf("enter string\n");
gets(a);
printf("%s", strrev(a));
}
```

Input : Rishitha
Output: ahtihsiR

strupr() : is the function used to read the string in lower case & convert the string into upper case.

Ex:
```
#include<stdio.h>
main()
{
char a[20];
printf("enter string\n");
gets(a);
printf("%s", strupr(a));
}
```

Input: Rishitha
output: RISHITHA

strlwr() : is the function used to read the string in upper case and convert the string into lower case.

Ex: #include <stdio.h>

```
main()
{
char a[20];
printf("enter string\n");
gets(a);
printf("%s", strlwr(a));
}
```

Input: RISHITHA

Output: rishitha

Strcmp(): is the function used to compare two strings and return 0 with if both the strings are equal other wise it returns the value of ascii difference of their first difference characters.

Ex: #include <stdio.h>

```
main()
{
char a[20],b[20];
int z;
printf("enter string\n");
gets(a);
printf("enter second string\n");
gets(b);
z = strcmp(a,b);
printf("%d", z);
}
```

India

Indonesia

i-o ⇒ 105-111

= -6

but in devc++ the difference value will show as -1

Input: enter string India

enter second string Indonesia

Output: -1

Write a c program to check whether both the strings are equal or not equal.

```
#include<stdio.h>
main()
{
char a[20], b[20];
```

```
int z;
Printf("enter first string \n");
gets(a);
printf("enter second string \n");
gets(b);
z = strcmp(a,b);
if(z==0)
printf("equal");
else
printf("not equal");
}
```

strcmpi(): is the function used to compare two strings just like strcmp but the only difference here is it ignores case sensitivity. (upper case and lower case letters are same).

Ex: India
  India

output: 0

Strcat()-string concatenation: is the function used to join two strings.

Ex:
```
#include <stdio.h>
  main()
  {
  char a[20], b[20];
  Printf("enter string\n");
  gets(a);
  printf("enter second string \n");
  gets(b);
strcat(a,b);
printf("%s", a);
}
```

a = | I | n | d | i | a | '\0' | | | | |
1 2 3 4 5 6 7 8 9 10

b = | k | e | n | y | a | '\0' | | | | |
1 2 3 4 5 6 7 8 9 10

Input: India
  Kenya

o/p: Indiakenya

output: Indiakenya

Strcpy() : is the function used to copy second string into the first string. that is 1st string gets replaced with 2nd string.

Ex: #include<stdio.h>
int main()
{
char a[20], b[20];
printf("enter string\n");
gets(a);
printf("enter second string\n");
gets(b);
strcpy(a,b);
printf("%s", a);
}

Input: India
Kenya

output: Kenya

Strncmp() : is the function used to compare n no. of characters of both strings.

Ex:-

n=3;
Strcmp(a,b,n);

output: 0

In the above example the result is 0. Even though both the strings are not same. Reason is n=3. which compares only first three characters of both the strings.

Strncat() : is the function used to join 'n' characters of second string to the first string.

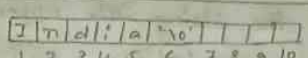
Ex: n=3;
Strncat(a,b,n);

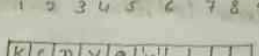
output: IndiaKen

Strncpy() : is the function used to replace the 'n' no. of characters of first string with 'n' no. of characters of second string.

Ex: n=3;

strncpy(a,b,n);





(kenya = 5
3 ⇒ ken
2 ⇒ ia
last digits of India
→ ia)

output: kenia (1st three characters should be same & last two character should change).
↓
ken

Write a c program to find the length of the string without using strlen() function.

Input: enter string
Indonesia

Output: 9

```
#include<stdio.h>
#include<string.h>
main()
{
char a[20];
int i,x=0;
printf("enter string\n");
gets(a);
for(i=0;a[i]!='\0';i++)
{
  x=x+1;
}
printf("%d",x);
}
```

Write a c program to read the string in lower case letters and convert the string into upper case letters without using strupr.

```
#include<stdio.h>
#include<string.h>
main()
{
char a[20];
int i;
printf("enter string\n");
gets(a);
for(i=0;a[i]!='\0';i++)
{
a[i]=a[i]-32;
}
printf("%s",a);
}
```

Input: enter string
india
output: INDIA

Write a C program to read the string into upper case letters and convert the string into lower case letters without using strlwr.

enter string
Input : INDIA
OUTPUT : India

```
#include<stdio.h>
#include<string.h>
main()
{
char a[20];
int i;
Printf(" enter string");
gets(a);
for(i=0; a[i]!='\0'; i++)
{
a[i]=a[i]+32;
}
Printf("%s", a);
}
```

Write a C program for displaying the following pattern.

```
#include<stdio.h>
main()
{
 int i,j;
for(i=65; i<=69; i++)
{
for(j=65; j<=i; j++)
 {
  Printf("%c", i);
 }
 Printf("\n");
}
}
```

Write a C program for displaying the following pattern.

```
#include<stdio.h>
main()
{
```

```
int i,j;
for(i='A';i<='E';i++)
{
for(j='A';j<=i;j++)
{
Printf("%c",j);
}
Printf("\n");
}
}
```

Write a C program to read the string (1 sentence) and count no. of words in a given string.

india is the best.

```
#include<stdio.h>
main()
{
 char a[30];
int i,x=0;
Printf("enter string");
gets(a);
for(i=0;a[i]!='\0';i++)
{
  if(a[i]==' ')
  {
   x++;
  }
}
 Printf("%d",x+1);
}
```

Input:
enter string
India is the best

output: 4

KL@123 4

** Write a C program to read the string and count no. of capital letters, small letters, digits, spaces, special symbols.

```
#include<stdio.h>
main()
{
```

if (a[i]>='A' && a[i]<='Z')
c++;

```
char a[30];
int i,d=0,c=0,sm=0,sp=0,z=0;
Printf("enter String");
gets(a);
for(i=0;a[i]!='\0';i++)
{
if(a[i]>=65&&a[i]<=90)
    {
     c++;
    }
else if(a[i]>=97&&a[i]<=122)
   {
   sm++;
   }
  else if(a[i]>=48&&a[i]<=57)
    {
    d++;
    }
  else if(a[i]==' ')
    {
     z++;
    }
  else
   {
   sp++;
  }
}
Printf("%d %d %d %d %d",c,sm,d,z,sp);
}
```

modified

The given input is got encrypted by replacing each character in a string with its respective next character. Display the updated string.

Input: ROSE

Output: SPTF

```
#include<stdio.h>
main()
```

```
{
char a[30];
int i;
Printf("enter string");
gets(a);
for(i=0; a[i]!='\0'; i++)
{
a[i] = a[i]+1;
}
Printf("%s", a);
}
```

Dynamic Memory Allocation (DMA): Allocating the memory during the run time of a program is known as dynamic memory allocation.

→ malloc()
→ Calloc()

are the functions used for Dynamic Memory Allocation.

Example programs using malloc().

→ Addition of two numbers using pointers.

```
#include<stdio.h>
main()
{
int *a,*b,p,q;
a= malloc(sizeof(int));
b= malloc(sizeof(int));
Printf("enter a,b");
Scanf("%d %d", &a, &b);
p=a;
q=b;
Printf("%d", p+q);
}
```

Write a C program to create an array using DMA, read the elements into array and display those elements.

```
#include<stdio.h>
main()
```

```
{
int *p,i,n;
printf("enter n");
scanf("%d",&n);
p=malloc(sizeof(int)*n);
printf("enter array elements");
for(i=0;i<n;i++)
scanf("%d",&p[i]);
for(i=0;i<n;i++)
printf("%d",p[i]);
}
```

p = calloc(sizeof(int),n);

Sum of array elements using Calloc: (DMA)

```
#include<stdio.h>
main()
{
int *p,i,n;
printf("enter n");
scanf("%d",&n);
p=calloc(sizeof(int),n);
printf("enter array elements");
for(i=0;i<n;i++)
scanf("%d",&p[i]);
for(i=0;i<n;i++)
x=x+p[i];
printf("%d",x);
}
```

Write a C program to reverse the string without using strrev().

```
#include<stdio.h>
main()
{
char a[80],c;
int i,j;
printf("enter name");
gets(a);
i=0;
j=strlen(a)-1;
```

```
While(i<j)
{
c=a[i];
a[i]=a[j];
a[j]=c;
i++;
j--;
}
Printf("%s",a);
}
```

Write a C program to join two strings without using strcat():

```
#include<stdio.h>
main()
{
char a[20], b[20];
int l,i;
Printf("enter first string");
gets(a);
Printf("enter second string");
gets(b);
l=strlen(a);
for(i=0;b[i]!='\0';i++)
{
  a[l]=b[i];
  l++;
}
a[l]='\0';
Printf("%s",a);
}
```

Write a C program to copy the second string to the first string without using strcpy function.

```
#include<stdio.h>
main()
{
char a[20], b[20];
int i,l;
```

```
Printf("enter first string");
gets(a);
printf("enter second string");
gets(b);
l=strlen(b);
for(i=0;b[i]!='\0';i++)
 {
   a[i]=b[i];
 }
 a[l]='\0';
puts(a);
}
```

Write a C program to read the string abbreviation and display its short form.

Ex: Indian Administrative Service - IAS.
Indian Airforce - IAF.

```
#include<stdio.h>
main()
{
char a[20];
int i;
Printf("enter the string");
gets(a);
for(i=0;a[i]!='\0';i++)
{
 if (i==0)
Printf("%c",a[i]);
 if(a[i]==' ')
printf("%c",a[i+1]);
}
}
```

Write a C program to check whether the given string is palindrome or not.

```
#include<stdio.h>
main()
{
```

```
char a[20], b[20];
Printf("enter the string");
gets(a);
strcpy(b,a);
strrev(b);
if (strcmp(a, b)==0)
printf("palindrome");
else
printf("not a palindrome");
}
```

Array of strings: Array of strings to be used in two dimensional character array.

Ex: char a[20][20];

In the above example 1st index talks about no. of strings we can read. Second index indicates the maximum range of each string.

* Though we have created 2 dimensional character array. We should use only the index of rows. that is no. of strings.

Ex: for(i=0; i<5; i++)
{
gets(a[i]);
}

	0	1	2	3	4	5	6	7	8	9	10	11
0	K	o	h	l	i	'\0'						
1	J	a	d	e	j	a	'\0'					
2	D	h	o	h	i	'\0'						
3	R	a	h	u	l	'\0'						
4	G	i	l	l	'\0'							

Write a C program to read 5 names and display those 5 names.

```
#include<stdio.h>
main()
{
char a[20][20];
int i;
Printf("enter strings\n");
for(i=0; i<5; i++)
{
fflush(stdin);
gets(a[i]);
}
for(i=0; i<5; i++)
{
Printf("%s\t", a[i]);
}}
```

Write a C program to read 5 cricketer names & display those names in a sorting order.

```
#include <stdio.h>
main()
{
Char a[20][20], c[20];
int i,j,n;
Printf(" enter n");
scanf("%d", &n);
Printf(" enter strings\n");
for (i=0; i<n ; i++)
{
fflush(stdin);
gets (a[i]);
}
for (i=0; i<n; i++)
{
for (j=0; j<n-i-1; j++)
 {
 if (Strcmp(a[j], a[j+1])>0)
 {
   Strcpy (c, a[j]);
   Strcpy (a[j], a[j+1]);
   Strcpy (a[j+1], c);
 }
}
}
Printf("Sorted Strings are \n");
for (i=0; i<n; i++)
{
Printf("%s \t", a[i]);
}
}
```

* a is for reading group of strings

if descending order:
if (strcmp(a[j], a[j+1])<0)

Structures : Structure is a Collection of elements of dissimmilar datatype.

The keyWord used for structure is Struct.

Example process for creation of structure

```
Struct student
{
 int id;
 char name[20];
};
```

structure name

Member variable of a structure

float marks;
} a; ← structure variable

In order to acess member variable of a structure we use dot operator (.).

Example: a.id
a.name
a.marks.

member variables

Write a C program to create a structure student with the fields id, name, marks and read the details of single student and display those details.

```
#include <stdio.h>
Struct Student
{
int id;
char name[20];
float marks;
} a;
main()
{
Printf("enter Student id, name, marks \n");
scanf("%d %s %f", &a.id, a.name, &a.marks);
Printf("%d %s %f\n", a.id, a.name, a.marks);
}
```

```
# include < stdio.h >
Struct Student
{
int id;
char name[20];
float marks;
};
main()
{
Struct Student a; → Struct : structure name; followed by variable name.
```

Write a C program to create a structure employee with the fields id, name, salary. Read the details of two employees and display those details.

```
#include<stdio.h>
Struct employee
{
int id;
char name[20];
float salary;
} a,b;
main()
{
Printf("enter first employee id, name, salary \n");
scanf("%d %s %f", &a.id, a.name, &a.salary);
printf("enter second employee id, name, salary \n");
```

input:
enter 1st employee id, name, salary
1
abc
102000
enter 2nd employee id, name, salary
2
bbc
9500

output: id name salary
1 abc 102000.000000
2 bbc 9500.000000

```
Scanf("%d %s %f", &b.id, b.name, &b.salary");
Printf("id \t name \t salary \n");
Printf("%d \t %s \t %f \n", a.id, a.name, a.salary);
Printf("%d \t %s \t %f", b.id, b.name, b.salary);
}
```

Break And Continue:

Break: Break keyword can be used in loops. If break keyword is used at particular condition then at that particular condition loop gets stop.

Ex:
```
#include <stdio.h>
main()
{
int i;
for (i=1; i<10; i++)
{
  if (i==7)
   break;
  printf("%d \n", i);
}
}
```

Output:
1
2
3
4
5
6

Continue: Continue is the keyword used to skip the loop statements at one particular condition.

Ex:
```
#include <stdio.h>
main()
{
int i;
for (i=1; i<=10; i++)
{
if (i==7)
Continue;
printf("%d \n", i);
}
}
```

Output:
1
2
3
4
5
6
8
9
10

Here number 7 is missing & later the loop will get increment upto i<=n.

if (i<=7)

Output: 8
9
10

* Command Line Arguments:

Command Line Arguments are based on two arguments values. They are

1) argc[ ]
2) argv[ ]

argc[ ]: argc[ ] counts number of input arguments + file location.

argv[ ]: argv[ ] gets the values in the form of strings.

In Dev C++ : Execute → Parameters → 10 20 30 40 → Enter values.

Ex:
```
#include <stdio.h>
main(int argc, char* argv[])
{
printf("%d\n", argc);
}
```
10 20 30 40

Output: 5

Reason: In parameters we have given 4 input values. So the result is 4+1 (file Location Count).

Ex:
```
#include <stdio.h>
main(int argc, char* argv[])
{
int i;
for(i=0; i<5; i++)
{
printf("%s\n", argv[i]);
}
}
```
10 , 20 , 30 , 40

C:\Users\DELL\Documents\zzz.exe

Output:
10
20
30
40

Reason: In parameters we have given 4 input values. So, the result is it will display the values.

Here 1st line of output is file location. In this index 0 is file location. from index 1 the values gets stored.

Structured Arrays:

Pakka Question: 5 or 6 in mid instead highest weightage

Structured Array is used to read the details of group of people like group of employes, group of cricketers. etc..

Ex:
```
struct emp
{
int id;
char name[20];
float salary;
} a[10]; → structure array
```

Write a C program to create a structure of employee with member variables id, name, Salary and read the details of 10 employees and display those details.

```
#include<stdio.h>
Struct emp
{
int id;
char name[20];
float salary;
} a[10];
main()
{
int i;
for(i=0;i<10;i++)
{
  Printf("enter id, name, salary \n");
scanf("%d%s%f", &a[i].id, a[i].name, &a[i].salary);
}
Printf("id\t name\t salary \n");
for(i=0;i<10;i++)
{
printf("%d\t %s\t %f\n", a[i].id, a[i].name, a[i].salary);
}
}
```

Write a C program to read the details of 5 employees (id, name, salary) and display the details of employee whose id is 30332.

```
#include <stdio.h>
Struct emp
 {
int id;
char name[20];
float salary;
} a[10];
main()
{
int i;
for(i=0;i<10;i++)
```

```
{
printf(" enter id, name, salary \n");
scanf(" %d %s %f", &a[i].id, a[i].name, &a[i].salary);
}
printf(" id \t name \t salary \n");
for(i=0; i<5; i++)
{
if (a[i].id == 30332)
printf("%d \t %s \t %f \n", a[i].id, a[i].name, a[i].salary);
}
}
```

Write Structure Search:

Write a C-program to read the details of 5 employees and search for the employee Navadeep using linear search technique.

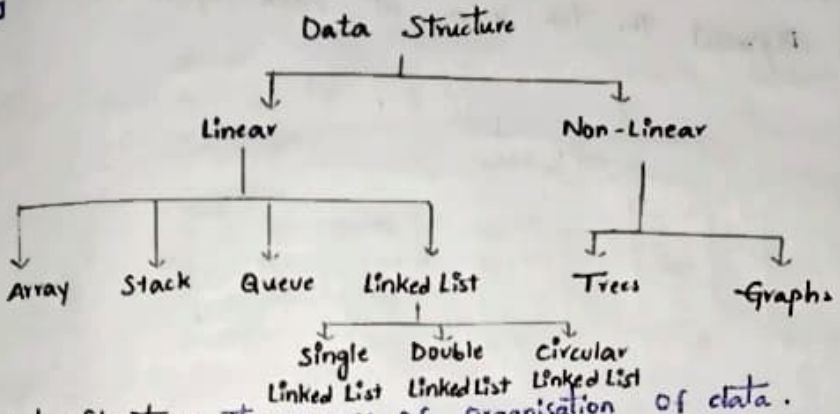
```
#include <stdio.h>
struct emp
{
int id;
char name [20];
float salary;
} a [10];
main()
{
int i, x=0;
for(i=0; i<5; i++)
{
printf(" enter id, name, salary \n");
scanf(" %d %s %f", &a[i].id, a[i].name, &a[i].salary);
}
printf(" id \t name \t salary \n");
for(i=0; i<5; i++)
{
  if(strcmp(a[i].name, "navadeep")==0)
        {
          x++;
          break;
        }
}
```

```
if (x==0)
printf("not found");
else
printf("found");
}
```

Write a C-program to read number of employees details and Sort those details based upon id using bubble sort.

```
#include<stdio.h>
struct emp
{
int id;
char name[20];
float salary;
} a[100],c;
main()
{
int i,j,n;
Printf("enter n");
scanf("%d",&n);
for(i=0; i<n;i++)
{
Printf("enter id, name, salary \n");
scanf("%d %s %f",&a[i].id, a[i].name, &a[i].salary);
}
for(i=0;i<n;i++)
{
for(j=0;j<n-i-1:j++)
{
if(a[j].id > a[j+1].id)
{
 c=a[j];
 a[j] = a[j+1];
 a[j+1] = c;
 }
}
}
```

```
printf("id \t name \t salary \n");
for(i=0; i<n; i++)
{
printf("%d \t %s \t %f \n", a[i].id, a[i].name, a[i].salary);
}
}
```



Data Structures: The process of organisation of data.
It is divided into two types.

* Linear data Structure
* Non-Linear data Structure.

Linear data structure: The organisation of data in linear order is said to be the linear data structure.

* Stack, Queue, Linked List comes under linear data structures.

Non-Linear data structure: The non-linear arrangement of data is said to be the Non-Linear data structure.

* Trees, Graphs comes under Non-Linear data structure.

Stack: Stack is a linear data structure where elements can be inserted and deleted from only one end. i.e Top.
The technique used in a stack is FILO (First In Last out) (or) LIFO (Last In First out)

Operations in a stack: Majorly there are two types of operations are there in stack. They are

1, push ( )
2, Pop ( )

Push(): The process of ~~inserti~~ insertion of elements into the stack is known as push operation.

Pop(): The process of deletion of elements from the stack is known as pop operation.

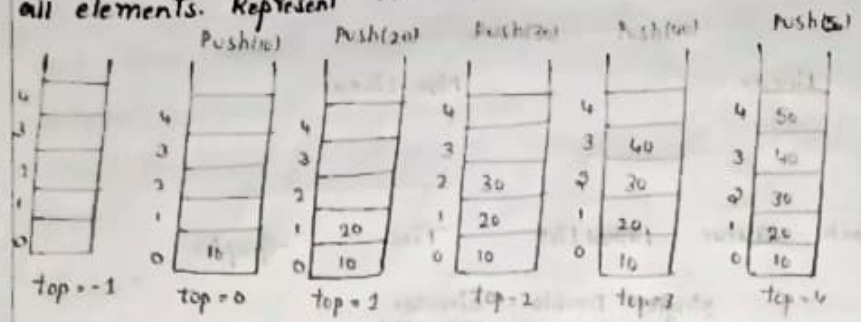
* Initially the value of top = -1. It gets incremented by 1 at each insertion.

If gets decremented by 1 at each deletion.

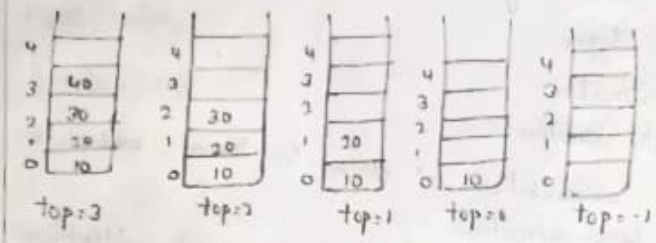
If top = -1 indicates Stack is empty.

If top = Stack size - 1 then it indicates stack is full.

Consider the elements 10,20,30,40,50 and perform push operation in a stack and also perform pop operation after insertion of all elements. Represent the top value at each operation.



Pop()



Write a menu driven program for the following operations on Stack. push(), pop(), display(), topp (or) peek().

```
#include<stdio.h>
int a[20], top = -1, size = 20;
main()
{
int ch;
while(1)
{
printf("enter 1 for push\n");
printf("enter 2 for pop\n");
printf("enter 3 for display\n");
printf("enter 4 for top\n");
printf("enter any other value for exist\n");
printf("enter ur choice\n");
scanf("%d", &ch);
switch(ch)
{
case 1: push();
```

```
break;
case 2: pop();
break;
case 3: display();
break;
case 4: topp();
break;
default : exit();
}
}
}
push( )
{
int n;
if (top == size-1)
{
Printf("cannot insert");
}
else
{
printf("enter n");
scanf("%d", &n);
top++;
a[top]= n;
}}
pop()
{
if (top == -1)
    {
     Printf("cannot delete");
    }
 else
   {
    top--;
   }
}
display( )
  {
```

```
int i,
for(i=top; i>=0; i--)
Printf("%d\t", a[i]);
}
```

```
topp() ---> topp()
 {
 printf("%d", a[top]);
}
```

Write the following functions or operations in a stack.

Push()

Pop()

display()

topp() (or) peek()

```
Push()
 {
  int n;
  if (top == size-1)
  {
  printf("cannot insert");
  }
  else
  {
  Printf("enter n");
  scanf("%d", &n);
    top++;
  a[top] = n;
  }
 }
```

```
Pop()
{
if (top == -1)
{
Printf("cannot delete");
}
else
{
 top--;
}
}
```

```
display()
{
int i;
for(i=top; i>=0; i--)
printf("%d\t", a[i]);
}

topp()
{
printf("%d", a[top]);
}
```

Queue(): In Queue we follow FIFO technique (first in first out) that means the element which is inserted first gets deleted first.

In a Queue both the ends are opened.

the end from which elements gets inserted is rear end.

the end from which elements gets deleted is front end.

Operations in a Queue(): majorly there are two types of operations in a Queue. (display is common. Insertion deletion is same.)

1) enqueue → process of insertion of elements into the Queue.

2) dequeue → process of deletion of elements from the Queue

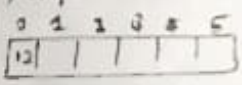
Consider the following operations on Queue() and represent the status of Queue() at each operation with the help of diagram. With their respective front and rear value.

→ we can replace alphabets

1) enqueue(12)
2) enqueue(45)
3) enqueue(64)
4) enqueue(25)
5) enqueue(68)
6) dequeue() → should delete 1st element. only front only incremented.
7) dequeue()
8) enqueue(90)

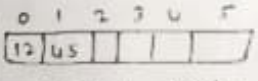
arraysize = 6

1) enqueue(12)



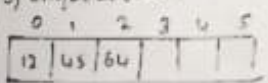
rear = 0 front = 0

2) enqueue(45)



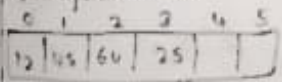
rear = 1 front = 0

3) enqueue(64)



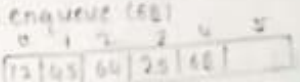
rear = 2 front = 0

4) enqueue(25)



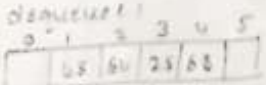
rear = 3 front = 0

5) enqueue(68)



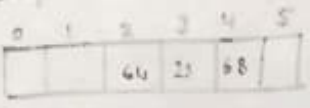
rear = 4 front = 0

6) dequeue()



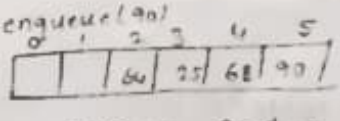
rear = 4 front = 1

7) dequeue()



rear = 4 front = 2

8) enqueue(90)



rear = 5 front = 2

Write a C program for the following operations on Queue( )

1, enqueue( )

2, dequeue( )

3, display( )

```
#include<stdio.h>
int a[20], front = -1, rear = -1, size = 20;
main()
{
int ch;
while(1)
{
Printf(" enter 1 for enqueue\n");
Printf(" enter 2 for dequeue\n");
printf(" enter 3 for display\n");
Printf(" any other value for exist\n");
Printf(" enter your choice");
Scanf(" %d", &ch);
Switch(ch)
{
case 1: Enqueue( );
   break;
case 2: dequeue( );
    break;
 case 3: display( );
    break;
default: exit(1);
}
}
}
enqueue( )
{
int n;
Print-f(" enter n value");
.scanf(" %d", &n);
if(rear = = size-1)
 {
```

```
printf("cannot insert");
}
else if (front==-1 && rear==-1)
   {
   rear=0;
   front=0;
   }
else {
   rear++;
   }
   a[rear]=n;
}
dequeue()
{
if (front==-1 && rear==-1)
{
printf("cannot delete");
}
else if (front==rear)
{
front=-1;
rear=-1;
}
else
{
 front++;
}
}
display()
 {
  int i;
if (front==-1 && rear==-1)
{
printf("cannot display");
}
for (i=front ;i<=rear; i++)
 {
 printf("%d\t", a[i]);
}}
```

String handling function:

Strstr(): is used to find the substring in a string.

Write a c program to read the string and substring and display the output from the first occurence of substring.

```
#include<stdio.h>
main()
{
 char a[20], b[20], *p;
printf("enter the first string\n");
gets(a);
printf("enter the sub string\n");
gets(b);
   p = strstr(a,b);
   Printf("%s", p);
}
```

Structure pointer: Structure pointers can also help us to access the member variable of a structure but for structure pointers we need to allocate the memory using malloc or calloc. Instead of dot operator(.) we go with arrow mark (->) to access the member variables of a structure using structure pointer.

Write a c program to create a structure student with the fields id, name, marks and read the values and display them using structure pointer.

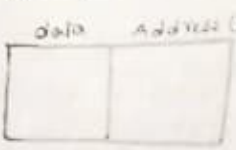
```
#include<stdio.h>
Struct students
{
 int id;
 char name[20];
 float marks;
}*a;
main()
{
a = malloc(sizeof(struct students));
Printf("enter id,name, marks\n");
Scanf("%d %s %f", &a->id, a->name, &a->marks);
Printf("id\t name\t marks\n");
Printf("%d \t%s\t %f", a->id, a->name, a->marks);
}
```

Write a C program to create a structure student with the fields id, name, marks & access the details in the main function using structure pointer. pass the same details to the function klul() using parameters and display those details with the help of function klul().

```
#include <stdio.h>
struct student
{
int id;
char name[20];
float marks;
} *a;
main()
{
a= malloc(sizeof(struct student));
Printf("enter id, name, marks\n");
scanf("%d %s %f", &a->id, a->name, &a->marks);
klul(a);
}
klul(struct student *a)
{
printf("id\t name\t marks\n");
printf("%d \t %s\t %f", a->id, a->name, a->marks);
}
```

Linked List: In Linked List all the nodes are connected with each other. Where each node contain 2 parts.

1) Data part
2) Address part (Address of next node).



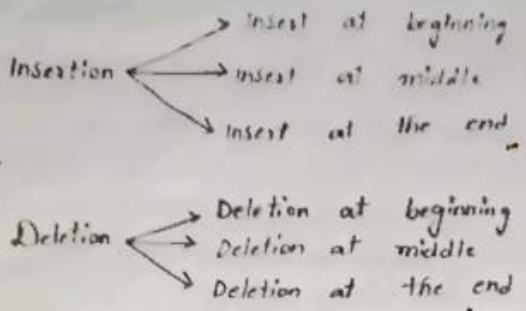
Node

These are 3 types of Linked List are there.

1) Single Linked List (SLL)
2) Double Linked List (DLL)
3) Circular Linked List (CLL)

Operations in a Linked List: These are two types of operations in a Linked List.

1) Insertion
2) Deletion



Write a Create function using single Linked List.

```
Create()
{
newnode = malloc (size of (struct node));
Printf(" enter n");
Scanf(" %d", & newnode → n);
if (start = = NULL)
{
Start = newnode;
newnode → next = NULL;
}
else
 {
  ptr = start;
  While(ptr → next ! = NULL)
  {
     ptr = ptr → next;
  }
   ptr → next = newnode;
   newnode → next = NULL;
}}
```

→ Write a function in Display Operation using Single Linked List.

```
display()
{
pta = start;
While (pta ! = NULL)
  {
    Printf(" %d \t ", pta → n");
        pta = pta → next;
  }
}
```

Write a function for Counting number of nodes in a single Linked list.

```
count()
{
  int x=0;
  ptr=start;
  while(ptr!=NULL)
  {
    x++;
   ptr=ptr->next;
  }
printf("%d\n",x);
}
```

Write a function for inserting new node at beginning using single Linked List.

```
insert_at_beginning()
{
newnode=malloc(sizeof(struct node));
Printf("enter n");
Scanf("%d",&newnode->n);
newnode->next=start;
start=newnode;
}
```

Write a function for inserting new node at end using single Linked List.

```
Insert_at_end()
{
newnode=malloc(sizeof(struct node));
Printf("enter n");
Scanf("%d",&newnode->n);
Ptr=start;
While(ptr->next!=NULL)
{
  Ptr=ptr->next;
}
Ptr->next=newnode;
newnode->next=NULL;
}
```

Write a function to delete a node at beginning using SLL.

```
delete_at_beginning()
{
 ptr = start;
 start = start -> next;
 free(ptr);
}
```

Write a function to delete a node at the end using SLL.

```
delete_at_end()
{
 Ptr = start;
 While(ptr -> next != NULL)
 {
  Pre = ptr;
  Ptr = ptr -> next;
 }
 Pre -> next = NULL;
 free(ptr);
}
```

Write a function to search for the key element using SLL.

```
Search_element()
{
 int key, x=0;
 Printf(" enter key \n");
 scanf(" %d", &key);
  ptr = start;
 While(ptr != NULL)
 {
  if (ptr -> n == key)
    {
      x++;
      break;
    }
   ptr = ptr -> next;
}
   if(x == 0)
  Printf(" not found\n");
  else
```

```
Printf(" found\n");
}
```

Write a function to find maximum element in SLL.

```
maximum_element_sll()
{
 int max;
 ptr = start;
 max = ptr->n;
 While (ptr != NULL)
 {
  if (ptr -> n > max)
  {
  max = ptr -> n;
  }
 ptr = ptr -> next ;
 }
 Printf (" %d \n", max);
}
```

Write a function to display the elements in reverse order.

```
reverse(struct node *ptr)
{
if (ptr != NULL)
 {
reverse(ptr -> next);
Printf(" %d ", ptr ->n);
  }
}
```

Write a function to sort the elements in a SLL.

```
Sort()
{
 int c;
for (ptr = start; ptr -> next != NULL; ptr = ptr -> next)
{
for (post = ptr -> next ; post != NULL; post = post -> next)
{
 if (ptr -> n > post -> n)
{
```

```
        c->pta->n;
        pta->n=post->n;
        post->n=c;
       }
     }
}
}
```

Write a function to insert a node at specific position (middle) Using SLL.

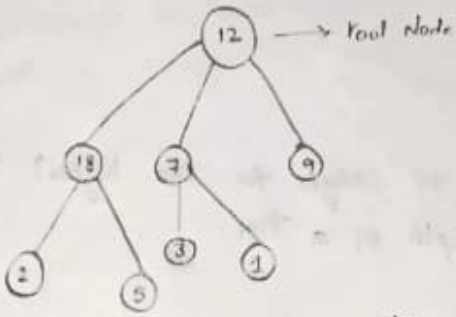
```
insert_at_middle()
{
    int x=1, pos;
   newnode = malloc(sizeof(struct node));
   printf("enter n");
    scanf("%d", &newnode->n);
    printf("enter position number\n");
    scanf("%d", &pos);
    ptr = start;
    while(x! = pos)
      {
        pre=pta;
        pta = pta->next;
        x++;
      }
   pre->next = newnode;
  newnode->next = pta;
}
```

Write a function to delete a node at specific position (middle) using SLL.

```
delete_at_middle()
{
 int x=1, pos;
printf("enter position number for deletion \n");
scanf("%d", &pos);
pta = start;
while(x! = pos)
    {
```

```
        pre = ptr;
        ptr = ptr->next;
        x++;
    }
pre->next = ptr->next;
  free(ptr);
}
```

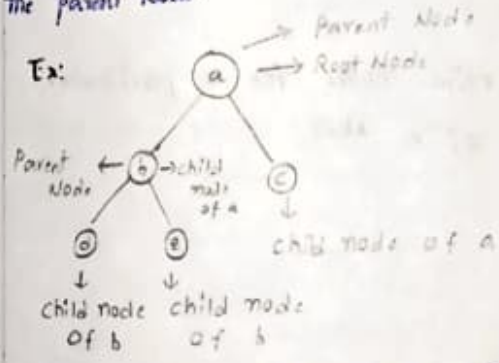
Trees: A tree is a non-linear data structure where the nodes are arranged in hyrarical manner.



Root Node: The starting node in a tree is said to be the root node.

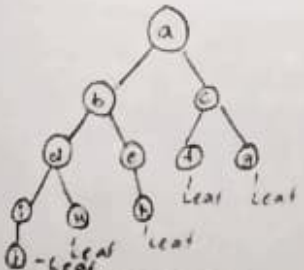
Parent Node: The node which is having children is said to be the parent Node.

Ex:



Siblings: child nodes having a common parent node are said to siblings.

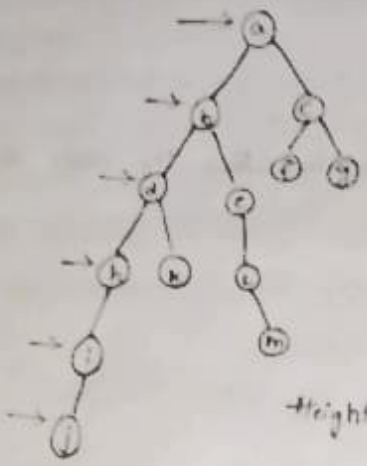
Leaf Node: Nodes which doesn't have even a single child node is said to Leaf Node (or) Node with zero child nodes are said to be Leaf Node.



-Height of a Node:

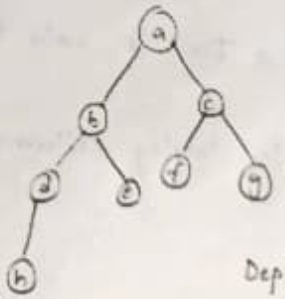
Height number of levels from Root node is said to be

-Height of node.



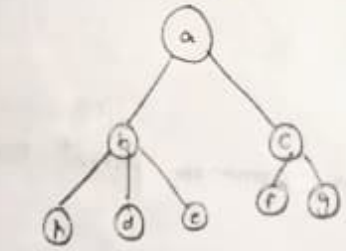
Height = 6

Depth of a Tree: Number of edges for the highest path of a tree is said to be depth of a tree.



Depth = 3.

Degree of a Node: Number of child nodes for a particular node is said to be a degree of a Node.



Degree of b = 3

Degree of c = 2

Types of Trees:

1, Binary Tree

2, Binary Search Tree

3, AVL Tree

4, B - Tree

5, Splay Tree

6, Red Black Tree

Binary Tree: In a binary tree each node must contain almost two child nodes.

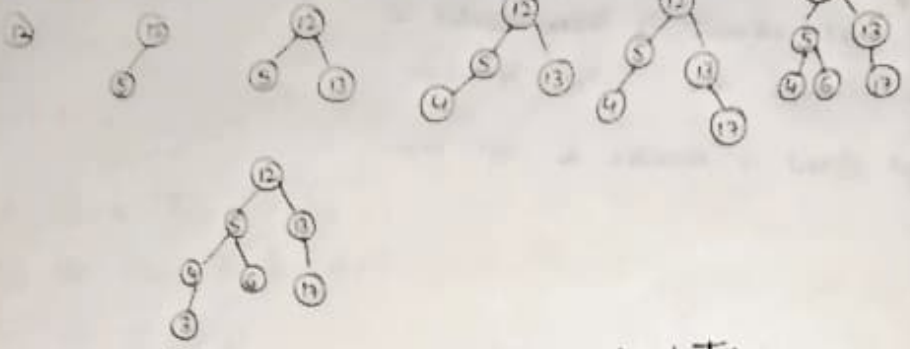
Binary tree

not a binary tree

Here b having 3 child nodes

Binary Search Tree: In binary search tree first element will be considered as a root node then the next elements will be inserted with the comparision of parent & root nodes. Smaller elements are inserted at the left side and larger elements are inserted at right side.

Ex Consider the elements 12, 5, 13, 4, 17, 6, 3 and construct binary search Tree.



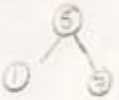
Traversal Techniques for binary tree and binary search Tree.

1) InOrder
2) Pre-order
3) Post-order

In-order: In this technique left node will be visited first then parent Node then right node.

LNR

L → Left
N → parent
R → Right



Inorder: 1-5-7

Pre-Order: In this technique parent node will be visited first then left node then right node.

NLR

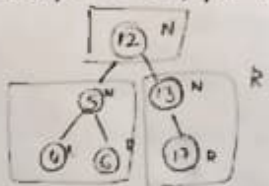
Pre-order: 5-1-7

Post-Order: In this technique left node will be visited first then right node and then parent node

LRN

Post-order 1-7-5

Consider the following tree & represent the order of element using In-order, post-order, pre-order



Inorder: 4-5-6-12-13-17

Preorder: 12-5-4-6-13-17

Postorder: 4-6-5-17-13-12

Graphs:

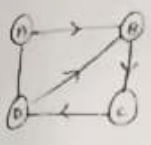
A graph is a non-linear data structure where the nodes are connected with each other with the help of their edges.

Types of Graphs:

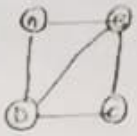
1, Directed Graph

2, Undirected Graph

Directed Graph



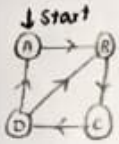
Undirected Graph



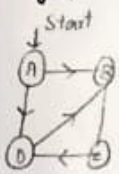
Cyclic and Acyclic Graphs

Cyclic graph is a directed graph where the direction from the starting node must lead its end to the same node (Starting & ending are same)

Eg:



Acyclic graph



(Starting & ending are not same)