Aakash kumar

1ST SEM

PPS

LUCKY

SPIRAL

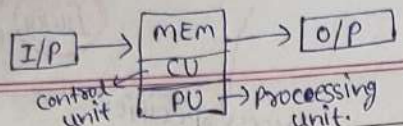
NOTEBOOK

for bright future....

COMPUTER

↓

Calculate



Lucky Date Page

ASCII → 128 → American standard code for Information interchange.

EBCDIC → 256 → Extended Binary coded Decimal Interchange code.

C-Programming

⇓

AT & T Lab california

→ Denis Ritche

Binary Number system:

$45 = 2^5 \times 1 + 2^4 \times 0 + 2^3 \times 1 + 2^2 \times 1 + 2^1 \times 0 + 2^0 \times 1$

$= 32 + 16 \times 0 + 8 + 4 + 2 \times 0 + 1$

$= 45$

N	N/2	N%2
10	5	0

$45 = (101101)_2$

Decimal to Binary

N	Quotient (N/2)	Remainder (N%2)
45	22	1
22	11	0
11	5	1
5	2	1
2	1	0
1	0	1

↑ Reverse.

$(101101)_2 = 45$

↓

Redix

BCD ⇒ Binary coded decimal.

Binary to Decimal

101101

$2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$ ⇒ $32 \times 1 + 16 \times 0 + 8 \times 1 + 4 \times 1 + 2 \times 0 + 1 \times 1 \Rightarrow 45$

Operators in C

① Arithmetic operators:- operators which generally performs basic operation like, addition, subtraction, multiplication, division etc.

→ list of Arithmetic operators.

ⓐ + → Addition → ex: a+b
ⓑ - → subtraction → ex: a-b
ⓒ * → multiplication → ex: a * b
ⓓ % → modulas → ex: a % b.

② Relational operators:- These types of operators are generally used to stablished the relation blw two variables.

→ example:- if a is equal to b.
if a is greater than b etc.

ⓐ == (equal to) → a is equal to b.
ⓑ != (Not equal to) → a is Not equal to b (a≠b)
ⓒ > (greater than) → a is greater than b (a>b)
ⓓ < (less than) → a is less than b (a<b).
ⓔ >= (greater ~~than~~ equal to) → a is greater than equal to b.
ⓕ <= (less than equal to) → a is less than equal to b.

③ logical operator:- these types of operators are used to calculate the value based on bits.

→ ex:- 0 AND 1 ⟶ 0
0 OR 1 ⟶ 1

Lucky Date __/__/____ Page _____

ⓐ AND (&&)

Truth table.

A	B	A*B
0	0	0
0	1	0
1	0	0
1	1	1

ⓑ OR (||)

A	B	AVB
0	0	0
0	1	1
1	0	1
1	1	1

ⓒ NOT (~, !)

A	~A
0	1
1	0

④ Increment & decrement operator

→ These operators are used to increase (↑) or decrease the value of a variable as per requirment.

ex:- a++ , b++ , ++a

Increment operator

Pre increment

→ variable should be increase 1st then Print the value.

→ increase then Print

→ ++a , ++b

Post increment.

→ variable should be Print First then increase the value.

→ Print then increase

→ a++ , b++

Lucky Date Page

Decrement operator

Pre decrement
→ First decrease then print
→ --a, --b

Post decrement
→ First Print. then decrease.
→ a--, b--

ex:
```
#include<stdio.h>
int main()
{
  int a=5, b=6;
  Printf("%d \n", a++);
  Printf("%d \n", ++a);
  Printf("%d \n", --a);
  Printf("%d \n", b--);
  return 0;
}
```

(5) Shift operator.
↳ these operator are used to shift the bit of particular variable either in left side or right side.

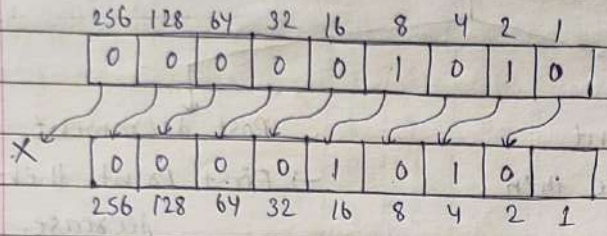
Types of shift operator:
↳ (a) left shift
(b) Right shift

(a) left shift
a=10
a<<1

(b) Right shift
a=15
a>>2

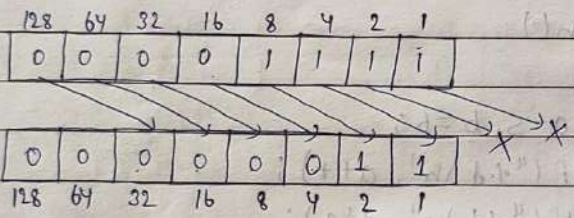
Lucky Date / / Page

* left shift



∴ a = 20

* Right shift



a = 3

⑥ Assignment operator.

↳ These types of operators are generally used to store the value of ~~the~~ one variable to another variable From right to left.

↳ ex - a = b

↳ value of b will assign to a

a = 5

b = 6

a = b;

~~6~~ 6

a b

① =, a = b → value of 'b' assign to variable a.

② +=, a += b ⇒ a = a + b;

③ -=, a-=b ⇒ a=a-b;
④ *=, x*=y ⇒ x=x*y;
⑤ /=, x/=y ⇒ x=x/y;
⑥ %=, P%=9 ⇒ P=P%9;
⑦ <<=, a<<=b ⇒ a=a<<b;
⑧ >>=, a>>=b ⇒ a=a>>b;

Ex:-

```
#include <stdio.h>
int main()
{
  int a, b;
  Printf ("Enter the value of a and b");
  Scanf("%d %d", &a, &b);
  a* = b;
  b%. = a;
  printf (" value of a is %d \n", a);
  printf (" value of b is %d \n", b);
  return 0;
}
```

→ Decesion control statements:

① if
② if - else
③ if - else if
④ Nested
⑤ switch

① if statement

↳ syntex

```
if (condition)
{
    statement 1;
    statement 2;
    :
}
```

```
if (condition)
    statement;
```

Que WAP to check the given no. is odd or even?

```
#include<stdio.h>
int main()
{
 int a;
 printf("Enter any +ve integer");
 scanf("%d", &a);
 if(a%2==0)
 {
  printf("%d is even", a);
 }
 if(a%2!=0)
 {
   printf("%d is odd", a);
  }
  return 0;
 }
```

Lucky Date Page

Flow chart

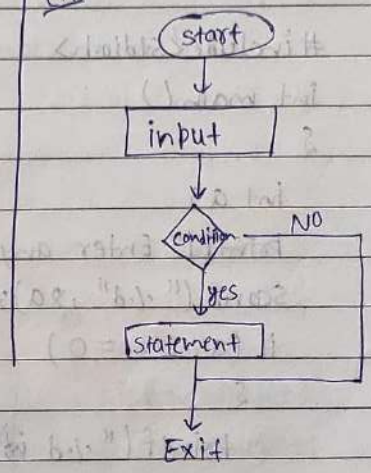
→ start/end

→ input

→ decesion

→ Process.

ex:-



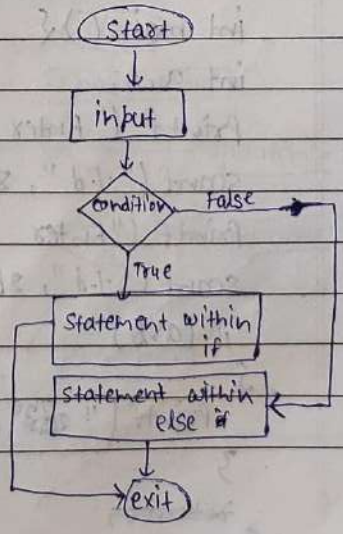
tex

```
if (condition)
{
 Statement 1;
 statement 2;
}
else
{
 Statement 1;
 statement 2;
}
```

or

```
└→ if (condition)
      statement;
   else
      statement;
```

EX:→ Flow chart.



Date ___/___/___
Page _____

Que WAP to Find the given no. is odd or Even?

```
#include<stdio.h>
int main()
{
  int a;
  printf(" Enter any +ve integer");
  scanf ("%d", &a);
  if (a%2==0)
    {
      printf("%d is even", a);
    }
  else
      printf("%d is odd", a);
  return 0;
}
```

Que WAP to Find the greatest number between two numbers?

```
#include<stdio.h>
int main(){
int a,b;
Printf ("Enter 1st Number");
scanf("%d", &a);
Printf ("Enter 2nd Number");
scanf("%d", &b);
if (a>b)
{
  Printf("The Greater Number is %d", a);
}
return 0;
```

```
else
  {
    Printf(" The greatest number is %d", b);
  }
return 0;
}
```

③ else if ladder

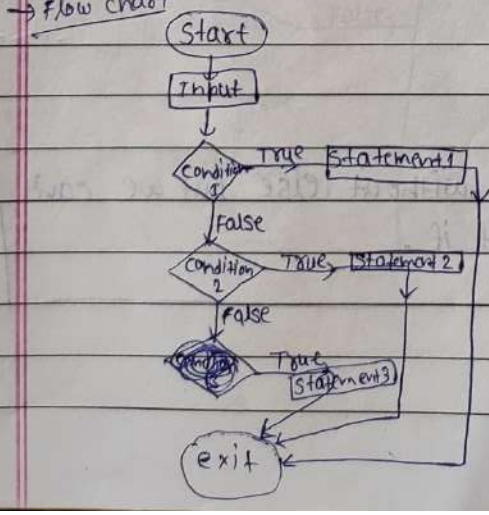
↳ syntex

```
if (condition 1)
  {
    Statement 1;
  }
else if (condition 2)
  {
    Statement 2;
  }
else
  {
    Statement 3;
  }
```

or

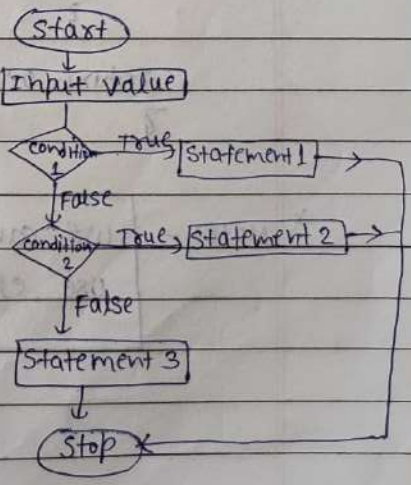
```
if (condition 1)
    statement1;
else if (condition2)
    statement 2;
else if(condition3)
    statement 3;
else if (condition 4)
    statement 4;
else
    statement 5;
```

→ Flow chart



Flow chart

→



Date ___/___/___
Page _____

Ques WAP to Find greatest no. among three nos using else if ladder.

```
#include<stdio.h>
int main()
{
   int a,b,c;
   Printf("Enter any three Number");
   scanf("%d, %d %d", &a, &b, &c);
   if (a>b && a>c)
   {
     Printf("%d is greatest than b and c; a);
   }
   else if ( b>a && b>c);
   {
     Printf("%d is greatest than a and c; b);
   }
   else
     {
       Printf("%d is Greatest", c);
     }
   return 0;
}
```

* Note:- we can use if without else but we can't use else without if.

④ Nested if

↳ syntex

```
if (condition 1)
{
    if (condition 2)
    {
        statement 1;
    }
}
```

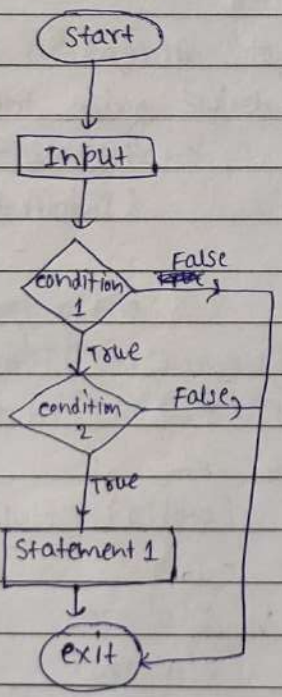
or

```
if (condition 1)
{
    if (condition 2)
    {
        statement 1;
    }
    else
    {
        statement 2;
    }
}
```

or

```
if (condition 1)
{
    if (condition 2)
    {
        statement 1;
    }
    else
    {
        statement 2;
    }
}
else
{
    statement 3;
}
```

→ Flow chart



Que WAP

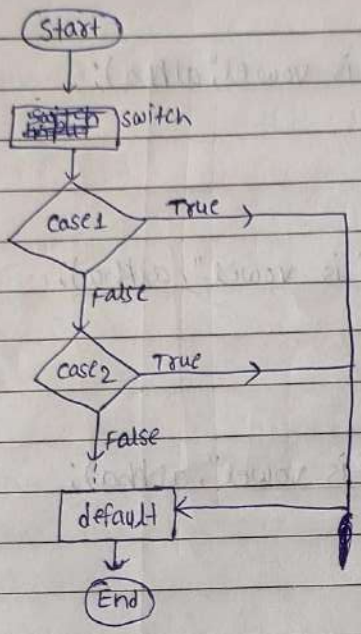
```
#include <stdio.h>
int main()
{
  if (ticket == 35)
  {
    if (Fish house ticket == 10)
    {
      Printf ("Now you go Fish house");
    }
  else
    {
      Printf ("Now you go Taramandal");
    }
return 0;
}
```

⑤ Switch case

↳ syntex

```
switch (condition)
{
  case '1':
    statement 1;
    break;
  case '2':
    statement 2;
    break;
  :
  default:
      Statement n;
}
```

→ Flow chart



Que write a program to Find even character is vowel or constant using Switch case.

```
#include<stdio.h>
int main()
{
char alpha;
  Printf("Enter any Alphabate : ");
  scanf("%c", &alpha);

  switch(alpha)
  {
     case 'a':
     case 'A':
     Printf("%c is vowel", alpha);
     break;
```

```
case 'e':
case 'E':
    Printf("%f is vowel", alpha);
    break;
case 'i':
case 'I':
    Printf("%f is vowel", alpha);
    break;
case 'o':
case 'O':
    Printf("%f is vowel", alpha);
    break;
case 'u':
case 'U':
    Printf("%f is vowel", alpha);
    break;
default:
    Printf("%f is consonante", alpha);
return 0;
}
```

Lucky Date ___ ___ ___
Page _____

* Loops in 'C':

→ Types of Loops.

Entry controlled loop.

→ A type of loop which check condition first then execute the statement.

↳ ex:- For loop
while loop

Exit controlled loop.

↳ A type of loop which excute ~~first~~ the code at once and then check the condition.

↳ ex:- do-while loop

* For loop in 'C'.

↳ Syntance

```
for (initialization; condition; increment/decrement/update)
{
  // body of for loop;
}
```

or

```
for (i=0; i<10; i++)
{
  // body of loop;
}
```

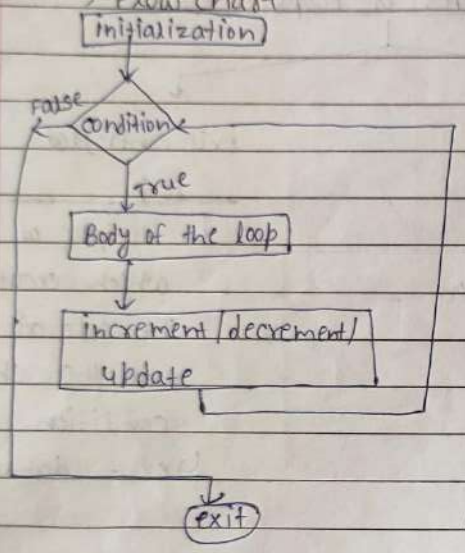
or

```
for(i=0; i<10; i++)
  // body of loop;
```

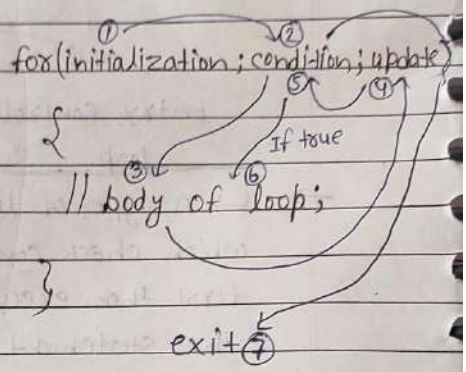
Lucky Date / / Page

* loops in 'c'

↳ Flow chart



↳ execution process of for loop:



Que write a program to print 10 natural numbers using for loop.

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

2 2×1
4 2×2
6 2×3
8 2×4
10 2×5
12 2×6
14 2×7
16 2×8
18 2×9
20 2×10

Lucky Date Page

Que write a program to print a table of any positive integer.

```
#include<stdio.h>
int main()
{
    int n;
    for(i=1 ; i<=10; i++)
    {
        printf("%d \n", 2*i);
    }
    return 0;
}
```

```
int n,i;
printf("Enter No: \n");
scanf("%d",&n);
for(i=1 ; i<=10 ; i++)
{
    Printf("%d \n", n*i);
}
return 0;
}
```

② while loop

↳ it is also a entry controlled loop.

↳ syntax

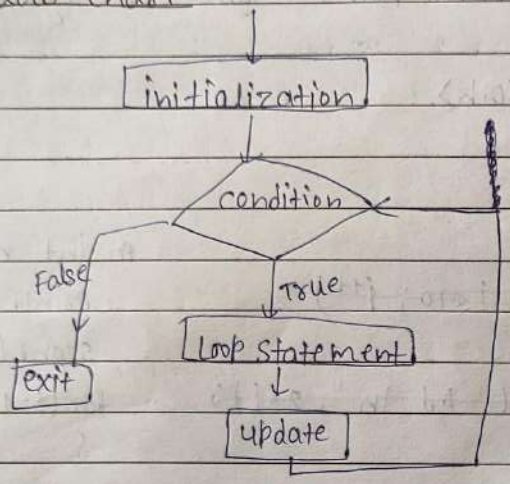
```
while (condition)
{
    // body of loop;
    increment/decrement;
}
```

or

```
initialization
while (condition)
{
    // body of loop;
    update;
}
```

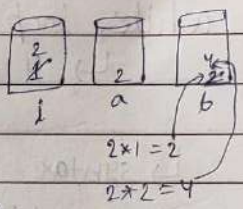
→ Flow chart.

→ Flow chart



Q:- write a program to print table of any +ve integer using while loop.

```
#include<stdio.h>
int main()
{
   int i=1,a,b;
   printf("Enter any +ve integer");
   scanf("%d",&a);
   while( i<=10 )
   {
    printf("%d \n", a*i);
    i++;
   }
   return 0;
}
```



③ do-while loop

↳ It is exit ~~ex~~ controlled loop.

↳ In do-while loop body of the loop will excute at least one if condition is false.

↳ syntex

↳ do
{
// body of loop increment/update;
} while (condition);

or

intialization
do
{
// body of loop;
update;
} while (condition);

→ Flow chart

Que write a program to print table of any +ve integer using do-while loop.

```
#include<stdio.h>
int main()
{
  int i=1,a,b;
  Printf("Enter any +ve integer");
  scanf("%d",&a);

  do
   {
     b=a*i;
     Printf("%d \n",b);
      i++;
    } while (i<=10);
  return 0;
}
```

Crate problem

```
#include<stdio.h>
int main()
{
 int a=0, b=5, c;
  c=a && ++b;
 Printf("value of c = %d", c);
 Printf("value of b = %d", b);
 return 0;
}
```

output

0,5

Lucky Date ___ ___ ___
Page _____

Que: write a program to print all even number between n to m.

```
#include<stdio.h>
int main()
{
  int n,m,i
  Printf("value n & m");
  Scanf("%d %d",&n,&m);
  for(i=n+1;i<m;i++)
  {
    if(i%2==0)
      Printf("Even");
  }
  return 0;
}
```

```
for(i=n;i<=m;i++)
{
  if(i%2!=0)
  Printf("odd");
}
return 0;
}
```

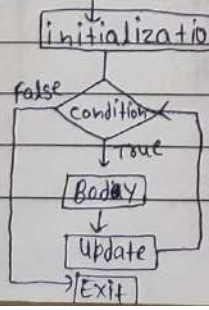
For Loop

① Entry controlled loop

② syntax

↳ for(initialization;condition;update)
{
// body of loop;
}

③ Flow chart



④ First check condition then excute the body of loop.

⑤ Body of the loop will not excute if condition

⑥ For is generally used when we know the no. of iteration.

Lucky Date __/__/__
Page

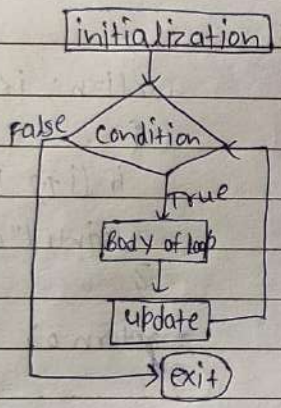
while loop

① it is also entry controlled loop

② syntax

```
while (condition)
{
// body of loop;
// update;
}
```

③ Flow chart



④ First check condition then excute the body of loop.

⑤ Body of the loop will not excute if condition false.

VVI ⑥ while is used when we don't know the no. of iteration.

do while

①

②

③

④

⑤ Body of the loop will excute at least once if condition is false.

⑥ do-while will used when we don't know the no. of iteration.

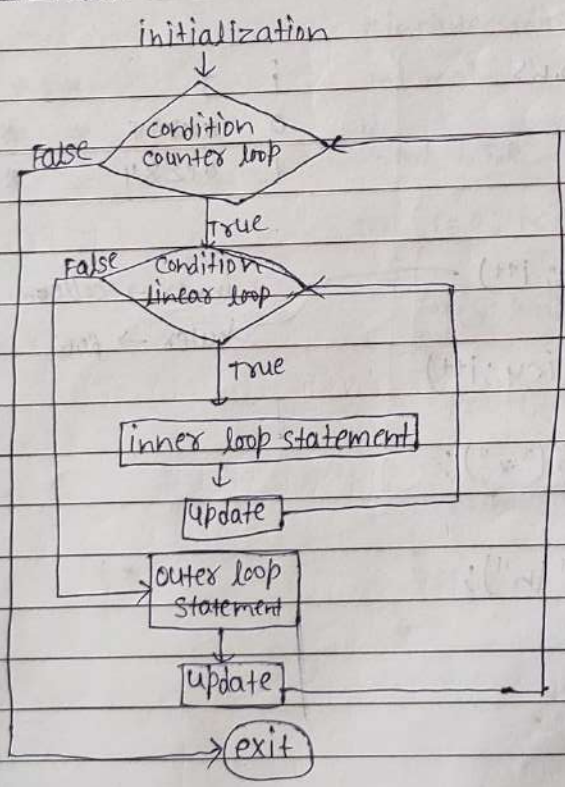
Nested Loops in C

① Nested For Loop.

↳ syntax

```
for(i=0; i<10; i++)
{
    for(j=0; j<10; j++)
    {
        // linear loop statement
    }
    // outer loop statement
}
```

→ Flow chart



Ex:-

```
for (i=0; i<3; i++)
{
    for(i=0; i<3; i++)
    {
        printf("welcome");
    }
    printf("ThanKS");
}
```

Que write a Program to Print Pattern:

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

o/p

```
* * * *
```

```
#include<stdio.h>
int main()
{
  Printf("* * * *");
return 0;
}
```

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

or

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

```
#include<stdio.h>
int main()
{
  int i,j;
for(i=0; i<2; i++)
  {
    for(j=0; i<4; i++)
    {
      Printf("*");
    }
    Printf("\n");
  return 0;
}
```

i	j	
0	0 1 2 3 4	* * * *
1	0 1 2 3 4	* * * *

inner → column
outer → Row

* *

Lucky Date ___ ___ ___
Page _____

Que WAP to Print Pattern.

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

```
#include<stdio.h>
Int main()
{
  int i,j;
  for(i=0; i<4; i++)
  {
    for(j=0; j<4; j++)
    {
      printf("* ");
    }
    Printf("\n");
  return 0;
}
```

Que

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

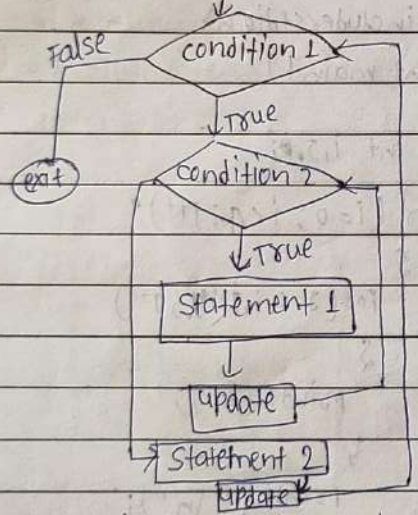
```
#include<stdio.h>
int main()
{
  int i,j,k;
for (i=0; i<4; i++)
  {
    For(j=0; j<i; j++)
    {
      Printf("*");
    }
    printf("\n");
  return 0;
}
```

② Nested while loop

syntex

```
while (conditions)
{
    while (conditions)
    {
        // statement 1;
        // update
    }
    // statement 2;
    // update;
}
```

→ Flow chart



③ Nested do-while loop

syntex

```
do
{
    do
    {
        // statement
        // update
```

10, 12, 18

Clearly if

} while (condition 2)
// update

} while (condition 1)

Que WAP to print the following output :

1
2 3
4 5 6
7 8 9 10

```
#include<stdio.h>
int main()
{
int i,j,n=0;
for(i=0;i<=4;i++)
{
for(j=0;j<=i;j++)
{
n=n+1;
printf("%d",n);
}
printf("\n");
}
return 0;
}
```

or n++

Lucky Date: Page:

operator	TYPE	ASSOCIAVITY
() [] . ->		left-to-right
++ -- + - ! ~ (type) * & sizeof	Unary operator	"
* / %	Arithmetic operator	"
+ -	Arithmetic operator	"
<< >>	shift operator	"
<<= >>=	Relational operator	"
== !=	Relational operator	"
&	Bitwise AND operator	"
^	Bitwise EX-OR operator	"
|	Bitwise OR operator	"
&&	Logical AND operator	"
||	Logical OR operator	"
?:	Ternary conditional operator	right to left
= += -= *= /= %= &= ^= |= <<= >>=	Assignment operator	right to left
,	Comma	left-to-right

Lucky Date ___ ___ ___
Page _____

Array in C

→ Need of array
↳ it is generally used to store the same data types.

↳ Before Array

```
int a=5;
int b=6;
int c=7;
```

↳ After Array

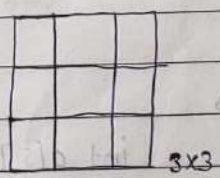
```
int a[3] = {5,6,7};
```

→ What is Array
↳ array is a collection of similar data types.
↳ it uses contigeous memory allocation.
↳ it allows uses to access memory in sequential manner.

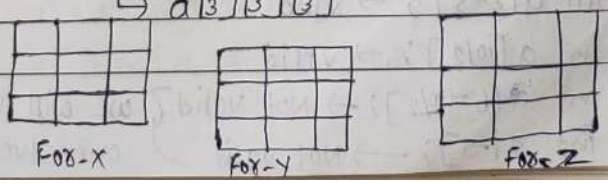
→ Types of Array.

① 1-Dimential array (1D array)
↳ memory representation
↳ int a[5] = | 2 | 4 | 3 | 2 | 1 |

② 2-D array
↳ memory representation.
↳ int a[3][3] →



③ 3-D array / multi dimentional array.
↳ memory representation
↳ a[3][3][3]



→ Array deceleration.

→ 1D-array → syntax

↳ data-type Name_of_array[size_of_array]

↳ ex:- int a[3];
float b[6];
char c[4];

→ 2D-array

↳ syntax

↳ data-type Name_of_array[row-size][column-size]

↳ ex:- int a[3][3];
float b[3][3];
char c[3][3];

→ 3D-array

↳ syntax

↳ data-type Name_of_array[x-size][y-size][z-size]

↳ ex:- int a[3][3][3];
float b[2][2][2];

Note 1)

int a[5]; → valid deceleration.

int a[]; → Error

↳ due to size not define.

int a[2*3]; → valid

int a[10/2]; → valid

int a[6=10/2]; → Not valid } we will use only +ve constant.

int a[-5]; → Not valid

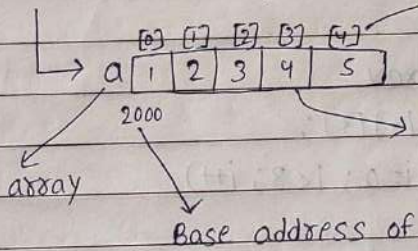
Lucky Date ___ ___ ___ Page _____

→ initialization of Array.

- **comple time initialization**
  - ↳ when an array initialize during it-comple time
  - ↳ ex:- `int a[2] = {1,2};`
- **Run time initialization**
  - ↳ when an array initialize during its run time.
  - ↳ cenerally uses is responsible to input array elements.

① compile time initialization

`int a[5] = {1,2,3,4,5};`



② `int arr[3] = {1,2,3,4,5};` → Error
↳ due to out of size.

③ `int arr[5] = {1,2,3};`
↳ Rest element will fill with zero.
↳ a = [1|2|3|0|0]

④ `int a[] = {1,2,3,4,5};` → valid
↳ a = [1|2|3|4|5]

⑤ `int a[5] = {10,2,-2,3,5};` → valid

⑥ `int a[5] = {10,2.3,6.5,2.1,3};` → Not valid.

→ 2D initialization

↳ int a[3][3] = {{1,2,3}, {4,5,6}, {7,8,9}};

↳ a[3][3]

1	2	3
4	5	6
7	8	9

3×3

→ Run time initialization.

↳ 1-D array

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

↳ 2-D Array

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

↳ 3-D Array

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

Que WAP to insert 5 elements in an array (1-D) having size equal to its elements.

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

```
// Print array elements.
for(i=0; i<5; i++)
 {
  printf("%d", a[i]);
 }
return 0;
}
```

Que WAP to insert and Print elements in 2-D array having size 3x3

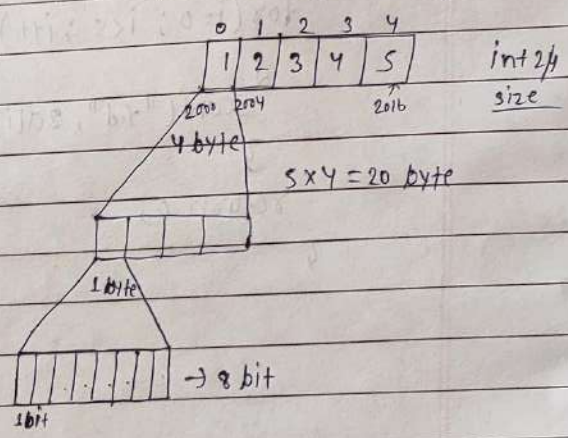
```
#include<stdio.h>
int main()
{
 int arr[3][3], i, j;
 Printf("Enter Array Element \n");
 for (i=0; i<3; i++)
  {
   scanf("%d", &arr[i]
   for (j=0; j<3; j++)
    {
     scanf("%d", &arr[i][j]);
    }
  }
```

```
// Print array Elements
for(i=0; i<3; i++)
 {
  for(j=0; j<3; j++)
   {
    printf("%d", arr[i][j]);
   }
 }
return 0;
}
```

→ memory allocation / representation of array.

↳ Array allow us to access array elements at any position.

int a[5] = {1,2,3,4,5};



→ access array element

B.A of index * size of data type.

Base Address

Ex:- 2000 + 1*4 = 2004

Ex:- 2000 + 4*4 = 20016

Que WAP to create two different ~~three~~ 3x3 matrix and add both matrix and store in 3rd matrix.

Soln

```
#include <stdio.h>
int main()
{
  int a[3][3], b[3][3], c[3][3];
```

```
int i,j;
Printf(" Enter elements of 1st matrix");
for(i=0; i<3; i++)
{
  for(j=0; j<3; j++)
  {
    scanf("%d", &a[i][j]);
  }
}

Printf(" Enter 2nd matrix data");
  for(i=0; i<3; i++)
  {
    for(j=0; j<3; j++)
    {
      scanf("%d", &b[i][j];
    }
  }

// Addition of matrix
  for(i=0; i<3; i++)
  {
    for(j=0; j<3; j++)
    {
      c[i][j] = a[i][j] + b[i][j];
    }
  }

// Print result matrix.
  Printf(" sum of matrix = \n");
  for(i=0; i<3; i++)
    {
      for(j=0; j<3; j++)
      { Printf("%d", c[i][j]");
      Printf("\n");
}
return 0;
}
```

Lucky Date Page

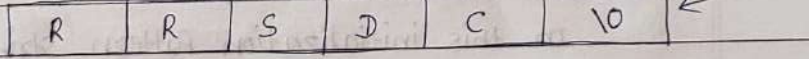
String in C

→ Character array

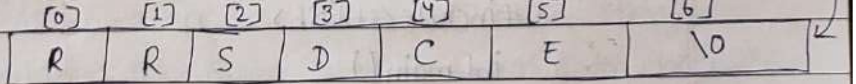
- It is array of character.
- Syntax
  - Data_type name_of_array[size_of_array];
  - char arr[6] = {'R', 'R', 'S', 'D', 'C', 'E'};
  - memory allocation

➤ String

- It is array of character but only different from character array, string ends with null value (\0).
- syntax
  - Data_type name_of_array[size_of_Array];
  - char arr[6] = {'R', 'R', 'S', 'D', 'C', '\0'};



  - char arr[7] = {'R', 'R', 'S', 'D', 'C', 'E', '\0'};



  - Size of string = Total Size − 1

→ Because last location occupied by null value (\0)

→ size of string means total no. of elements store in string.

Lucky Date __/__/__ Page _____

➤ Initialization of string

1. Compile time initialization :-

a. `char ch[] = "CSE";`

- size = 4

[0]	[1]	[2]	[3]
C	S	E	\0

b. `char ch[] = "Dept. of CSE";`

- size = 13

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]
D	e	p	t	.		o	f		C	S	E	\0

c. `char[4] = "CSE";`

[0]	[1]	[2]	[3]
C	S	E	\0

d. `char[5] = "CSE";`

[0]	[1]	[2]	[3]	[4]
C	S	E	\0	

2. Run time initialization :-

In this initialization pattern string will be initialize during run time.

Example :-

```
#include<stdio.h>
int main()
{
    char arr[20];
    printf("Input string :- ");
    scanf("%s", &arr);
    printf("output string :- %s", arr);
    return 0;
}
```

```
C:\Users\USER\OneDrive\Doc\string.c
Input string :- CSE
output string :- CSE
```

Note:- There is no need of for loop and ampersand (&) required to store elements in character array.

➤ Scanf and gets function

- Scanf function is used to store the string in to character array.
- Syntax:-

```
scanf("%s", &Name_of_Array);
```

- Example :-

```
#include<stdio.h>
int main()
{
    char arr[40];
    Printf("Input string:-");
    scanf("%s", &arr);
    Printf("output using scanf function:- %s", arr);
    return 0;
}
```

```
C:\Users\OneDrive\Desktop\Ranjan\string.exe
Input string:- RRSDCE
output using scanf function:- RRSDCE
```

➤ Drawback or limitations of scanf function

- Limitation #1 - Scanf function is not able to read white-space or blank space in string.
- Limitation #2 - Scanf function unable to check boundary or size of buffer. i.e buffer overflow Problem occur.
- ~~Example :- below example is unable to Print Dept. of CSE.~~

Date ___/___/___
Page _____

- Example :- below example is unable to print Dept. of CSE.

```
#include<stdio.h>
int main()
{
    char arr[20];
    Printf("Input string :- ");
    scanf("%s", &arr);
    Printf("output string :- %s", arr);
    return 0;
}
```

C:\Users\OneDrive\Documents\String1.exe
Input string :- Dept. of CSE
Output String :- Dept.

- To overcome white space or blank space problem we use gets function.

➤ gets function

- gets function able to read blank space of any string
- Syntax :-

```
gets(Name_of_String);
```

- Example :-

```
#include<stdio.h>
int main()
{
    char arr[40];
    printf("Input string :-"));
    gets(arr);
    Printf("output using gets function :- %s", arr);
    return 0;
}
```

C:\Users\USER\OneDrive\Desktop\string2.exe
Input string :- College of Engg Begusarai
output string :- College of Engg Begusarai

➤ Drawback or limitations of gets function

- Limitation#1- gets function unable to check boundary or size of buffer. i.e buffer overflow Problem occur.

- Example:-

```
#include<stdio.h>
int main()
{
  char arr[10];
  Printf("size of actual character array=%d \n", sizeof(arr));
  Printf("Input string:- ");
  gets(arr);
  Printf("output using gets functions:- %s ", arr);
  return 0;
}
```

```
c:\users\USER\oneDrive\Desktop\Ranjan\string3.exe  - □ x | CSE
size of actual character array = 10
Input string :- Dept.of computer science & Engineering.
output using gets functions :- Dept. of computer science &
                                Engineering
```

- In above example the actual size of character array is 10 but we are inserting 40 characters in character array arr[10]. Dues to this buffer overflow occur.
- To overcome the problem of buffer overflow we will use fgets function.

➤ fgets function in C

- Using fgets function, we can resolve the problem of scanf function and gets function i.e buffer overflow problem.
- Syntax:-

```
fgets(Name_of_array, size_of_array, stdin);
                                    ↳ standard input
```

- Example:-

```
#include<stdio.h>
int main()
{
    char arr[40];
    Printf("Input string:- ");
    fgets(arr, 40, stdin);
    Printf("output using fgets function:- %s", arr);
    return 0;
}
```

```
C:\Users\OneDrive\Aakash\string4.exe
Input string:- Dept. of CSE
output using fgets function:- Dept. of CSE
```

➤ Printf and puts function

- printf and puts function are generally used to print the string in console window.
- Only the difference is puts function already define with new-line function (\n).
- Example #1:- output using Printf function

```
#include<stdio.h>
int main()
{
    char ch1[40], ch2[40];
```

```
Printf(" Input string 1 :- ");
gets (ch1);
Printf(" Input string 2 :- ");
gets(ch2);
Printf(" output :- ");
Printf("%.s ", ch1);
Printf("%.s", ch2);
return 0;
}
```

```
c:|users|onedrive|desktop|string7.exe
Input string 1 :- welcome to college of Engineering
Input string 2 :- Begusarai
output :- welcome to college of Engineering Begusarai
```

- Example #2 :- output using puts function

```
#include<stdio.h>
int main()
{
    char ch1[40], ch2[40];
    Printf("Input string 1 :- ");
    gets (ch1);
    Printf("Input string 2 :- ");
    gets(ch2);
    Printf("Print using Puts function :- ");
    Puts(ch1);
    Puts(ch2);
    return 0;
}
```

```
Input string 1 :- welcome to college of Engineering
Input string 2 :- Begusarai
Print using Puts function :- welcome to college of Engineering Begusarai
```

Begusarai

Date ___/___/___
Page _____

- Note:- when we use the fgets function instead of the gets function, the printf function behaves similarly to the puts function.

- Example:-

```
#include<stdio.h>
int main()
{
    char ch1[40], ch2[40];
    Printf("Input string 1:- ");
    fgets(ch1, 30, stdin);
    Printf("Input string 2:- ");
    fgets(ch2, 30, stdin);
    Printf("output:- ");
    Printf("%s", ch1);
    Printf("%s", ch2);
    return 0;
}
```

```
C:\Users\OneDrive\Desktop\Ranjan\string6.exe
Input string 1:- Department of
Input string 2:- computer science & Engineering
output:- Department of
computer science & Engineering
```

Que write a C Program to calculate the length of string?

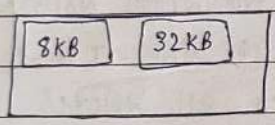
Lucky Date Page

Functions in C

① Reuseability :-

Without using function

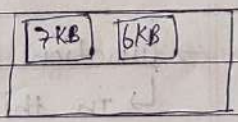
```
1 #include<stdio.h>
2 int main()
3 {
4 int a=8, b=6, c=0;
5 c= a+b;
6 Printf("%d", c);
7 return 0;
8 }
```



② memory managment :-

using function

```
1. #include<stdio.h>
2. int main()
3. {
4. sum();
5. Printf("Thanks");
6. return 0;
7. }
8. sum()
9. {
10. int a=8, b=6, c=0;
11. c= a+b;
12. Printf("%d", c);
13. }
```



Que what is function

↳ function is a group of statements which Perform specific task.

↳ function can be called multiple times in a single program.

↳ A function contains set of code to Perform some specific task.

Lucky Date ___/___/___ Page _____

→ Types of function

- Pre define function
  - ↳ Pre define function are also called libaraty function
  - ↳ ex:- printf(), scanf() etc.
- user-define function
  - ↳ It is define by user according to its requirements.
  - ↳ sum(), sub() etc.

→ Parts of function

↳ functions are generally 3 parts

- function decelaration
- function calling
- Function define.

→ function decelaration

↳ In this part we decelare the function for identification.

↳ syntax

↳ data_type Name_of_function (Parameters/arguments);

↳ Ex:

↳ int sum(int x, int y);

↳ void sum(void);

Lucky Date ___ ___ ___
Page _____

→ Function calling

↳ It is used to call the defined functions as per requirement.

↳ syntax

↳ Name_of_function(Arguments/ Parameters);

↳ ex:-

↳ sum();

↳ add(x,y);

→ function definition

↳ In this part we defined the function in details with its working methods.

↳ syntax:

↳ data_type Name_of_function(Arguments/Parameters)

{

// body of function;

return ;

}

Ex:-

```
#include<stdio.h>
int sum();  ---> function decelaration
int main()
{
  sum();  ---> function calling
  printf("Thanks");
  return 0;
}
int sum()  ---> function defination
{ int a=6, b=4, c=0;
  c=a+b;
  printf("%d", c);
  return 0;
}
```

Lucky Date __/__/__
Page _____

→ Types of function in C

↳ They are defined in 4 types

① function without arguments without return value.
② function with arguments without return value.
③ function without arguments with return value.
④ function with arguments with return value.

① without arguments without return value.

```
#include<stdio.h>
void sum(void);
void main( );
{
   sum ();
   Printf("\n Thanks");
}
void sum()
{
   int a=5 , b=5, c=0;
   c=a+b;
   Printf("sum=%d", c);
}
```

② with arguments without return value.

```
#include <stdio.h>
int sum (int a , int b);
int main()
{
int a=5 ;int b=5;
   sum(a,b);
   printf("\n Thanks");
```

Actual Parameter

Formal parameters.

```
int sum (int x, int y)
{
    int c=0;
    c=x+y;
    printf("sum = %d", c);
    return 0;
}
```

WAP to ~~Print~~ calculate simple interest using function (with arguments without return value)

```
#include<stdio.h>
int S.I(int, int, int);
int main()
{
    int p=1000, r=10, t=2;
    S.I(p,r,t);
    return 0;
}
int S.I (int x, int y, int z)
{
    int c=0;
    c = (x*y*z)/100;
    printf("S.I = %d", c);
    return 0;
}
```

③ without arguments with return value.

```
#include<stdio.h>
 int add( );

 int main()
 {
   int r=0;
    r= add();
    Printf("sum=%d",r)
    return 0;
  }
  int add(){
  int a=5, b=6;
    return a+b;
  }
```

④ with arguments with return value.

```
#include<stdio.h>
 int add(int, int);
 int main()
 {
    int a=5, b=6, c=0
    c= add(a,b);
    printf("sum=%d", c);
    return 0;
  }
int add(int x, int y)
 { int c=0;
    c=x+y;
    return c;
```

WAP to calculate the area of circle using function. with arguments with return value. where variables initialies during run time.

```
#include<stdio.h>
float area(int r);
int main()
{
  int r;
  area(r);
  printf("%f", area);
  return 0;
}
float area(int r)
{
  printf("Enter radius");
  scanf("%d", &r);
  float pi=3.14, a;
  a= pi*r*r;
  printf("Area is %f", a);
  return a;
}
```

WAP to check given number is prime or not using function.

WAP to ~~factorial~~ find factorial of any number. using function.

```
#include<stdio.h>
int ~~aa~~fact();

int main(){
 fact()
 return 0;
}
int fact(){
 int n,i;
 Printf("Enter number");
 Scanf("%d",&n);
 n=1;
 for(i=0;i<n;i++)
   {
     n=n*(n-i);
   }
 printf("%d",n);
 return 0;
}
```

marks[] = {1,2,3,4,5}
number = size of(marks) / sizeof(marks[0])
First element.

Lucky Date / Page

→ Passing array through function.

```
#include <stdio.h>
int student (int[]  );
int main()
{
  int marks[10];
  int i;
  Printf ("Enter array elements");
  for (i=0 ; i<n ; i++)
  {
    scanf ("%d", &marks[i]);
  }
  student (marks , n);
  return 0;
}
int student(int arr[] , int x)
{
  int i;
  for (i=0 ; i< x ; i++)
  {
    printf ("%d ", arr[i])
  }
  return 0;
}
```

n = size of (marks) / size of (marks[0])

Que WAP to insert 10 elements in an array & calculate sum of all elements using function.

```
#include<stdio.h>
int sum(int[] , int);
```

```
int main()
{
  int arr[10];
  int i;
  Printf("Enter array element");
  For(i=0; i<10; i++)
  {
     scanf("%d", &arr[i]);
  }
  sum(arr, 10);
  return 0;
}
int sum(int arr[], int x)
{
  int sum=0
     int i;
  for(i=0; i<10; i++)
  {
     sum = sum + arr[i];
  }
  Printf("sum is %d", sum);
  return 0;
}
```

H.W

Q1/ WAP to insert 10 element in an array and print even element and odd elements seperatily using function.

```
#include<stdio.h>
int even(int [], int);
int odd(int [], int);
int main()
{
  int n[10], i;
  Printf("Enter 10 Array elements");
  For(i=0; i<10; i++)
    {
      scanf("%d", &n[i]);
    }
  int even(int, 10);
  int odd(n, 10);
  return 0;
}

int even(int n[], int);
{
  int i;
  printf("even numbers");
    for(i=0; i<10; i++)
      {
        if(n[i] %2 ==0)
          {
            printf("%d", n[i]);
          }
      }
  return 0;
}
```

```
int odd (int n[], int y);
{
    int i;
    Printf ("odd numbers");
    for(i=0; i<10 ; i++)
    {
        if (n[i] % 2 != 0)
        {
            Printf("%d", n[i]);
        }
    }
    return 0;
}
```

Lucky Date ___ ___ ___
Page _____

Pointer in c

↳ Pointers are generally used to store the address of variables in program.

↳ for different date type pointer should be different.

Like- for int a, pointer will be int *p

→ for float b, Pointer will be float *p2

→ for char a, Pointer will be char *p3

↳ The size of pointer variable depends on compiler or system.

↳ Pointer is also called derived data type.

→ syntax

↳ data_type *Name_of_Pointer

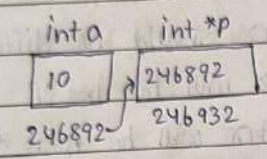
↳ Ex- int *a;
float *b;
char *c;

→ Pointer initialization

↳ int a=10;
float b=20.34;
char c='R';

int *p;
p=&a → Valid
p=&b → x (Not valid)
p=&c → x (Not valid)

```
int a=10;
int *p = &a;
```



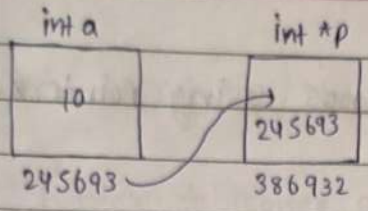
int *p ⟶ Pointer p store the address of variable a, where a is integer data type.

float *p1 ⟶ Pointer p1 store the address of variable, where variable data type is float.

char *c ⟶ Pointer c store the address of variable, where variable data type is char.

double *d ⟶ Pointer d store the address of variable, where variable data type is double.

```
#include<stdio.h>
int main()
{
    int a=10;
    int *p;
    p=&a;
    Printf("%d \n",a);
    Printf("%d \n",&a);
    Printf("%d \n", p);      printf("%d \n",&*p);
    Printf("%d \n", *p);
    Printf("%d \n",&p);
    return 0;
```



address of a = 245693	10 → a
address of *p = 386932	245693 → &a
value of a = 10	245693 → p
value of *p =	10 → *p
	386932 → &p
	245693 → &*p

→ Pointer Arithmatic

↳ Pointer can also be use as a arithmatic operation like variables.

↳ Ex- int *p1;
int *p2;

then *p1 + *p2 → valid
*p1 * *p2 → valid
*p1 − *p2 → valid.

↳ *p1 > *p2 → valid
*p1 ≤ *p2 → valid
*p1 && *p2 → valid
*p1++ → valid
*p2-- → valid.

Lucky Date ___/___/___ Page _____

Que WAP to add two no's using pointers.

```
#include<stdio.h>
int main()
{
  int a=5, b=6, c=0;
  int *p1=&a, *p2=&b, *p3=&c;
  *p3 = *p1 + *p2;
  printf("sum = %d", *p3);
  return 0;
}
```

Que WAP to Find greatest number ~~using~~ among three number using ~~functions~~ pointer.

```
#include<stdio.h>
int main()
{
  int a, b, c;
  Printf("Enter Three Numbers:");
  scanf("%d %d %d", &a, &b, &c);
  int *p1=&a, *p2=&b, *p3=&c;
  if(*p1>*p2 && *p1>*p3)
  {
    printf("Greatest Number:", *p1);
  }
  else if(*p2>*p1 && *p2>*p3)
  {
    Printf("Greatest Number:", *p2);
  }
  else
  {
    printf("Greatest Number:", *p3);
  }
```

→ Pointer to Pointer / Double pointer

↳ Pointer to pointer are Generally use to store the address of another Pointers.

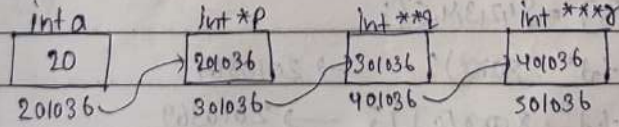
↳ EX- int a = 10;
int *p = &a;
int **q = &p;

Note:-

int a=10;
int *p=&a; ⟶ valid
int **q=&p; ⟶ valid
int ***r=&q; ⟶ valid
int **q=p; ⟶ valid
int *p=q; ⟶ valid
int *p=r; ⟶ valid
int *p=a; ⟶ invalid (X)
int **q=a; ⟶ invalid (X)
int ***r=a; ⟶ invalid (X)

EX:-

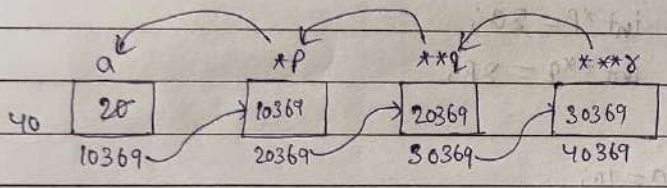
int a=20;
int *p=&a;
int **q=&p;
int ***r=&q;



Note:→ 3rd level Pointer r will store the address of 2nd level Pointer q.
→ 2nd level " q " " " " " 1st level Pointer p.
→ 1st level " p " " " value of variable a.

Ex int a=20;
int *p=&a;
int **q=&p;
int ***r=&q;

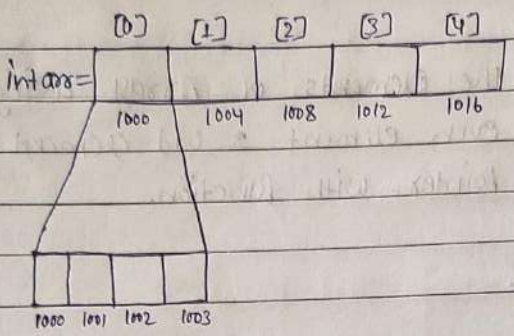
→ access or Edit the value of a by using r pointer.



((*(r))) = 40;
((*(30369)))
((20369))
*(10369)
a = 40

→ Pointer and array.
or
Pointer array / array pointer

```
#include<stdio.h>
int main()
{
  int arr[5] = {1,2,3,4,5};
  Printf("%d", &arr);      → 2010369
  Printf("%d", &arr[0]);   → 2010369
  return 0;
}
```




```c
#include<stdio.h>
int main()
{
  int arr[5], i;
  printf(" Enter array elements ");
  for(i=0; i<5 ; i++)
  {
    scanf("%d", &arr[i]);
    scanf("%d", arr+i);
  }
  printf("Array elements are ");
  for(i=0; i<5; i++)
  {
    printf("%d", arr[i]);
    printf("%d", *(arr+i));
  }
  return 0;
}
```

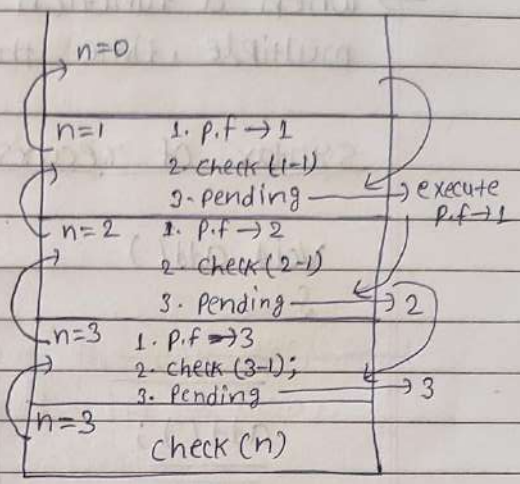
int a=5
int *p=&a
printf("%d", *p);
↳ 5

Value Address

Palindromic number

Date ____ ____ ____
Page ____

```
#include<stdio.h>
int check(int);
void main()
{
   int n=3;
    check(n);
}
int check(int n)
{
   if(n<1)
     return 0;
   else
    {printf("%d",n);
     check(n-1);
     printf("%d",n);
    }
}
```



output: - 321123

Recursion in C

→ when a function call itself or other functions, multiple time then it is call recursion.

Syntax of recursion.

```
Void add()
{
   ____
   ____
   add();
   ____
}
```

```
int main()
{
   ____
   ____
   add()
   ____
}
```

Example

Syntax:

```
int main()
{
   // statement;
   array();
   // statement;
}

int array()
{
   // statement;
   array();
   // statement;
}
```

Array of function
Array of pointer.
Array of string.
function of pointer.
pointer of function

Que wap to ~~Print~~ Find out the day ~~definicat~~ by a number using pointer & Array.

```
#include<stdio.h>
int main()
{
  char A[7] = {"S","M","T","W","TH","F","SAT"};
  int n;
  printf("Enter number");
  scanf("%d",&n);
  int *p = &n;
  if(*p==1)
  {
    for(i=0;i<1;i++)
    {
      printf("%c",A[i]);
    }
  }
  else if(*p==2)
  {
    for(i=1;i<2;i++)
    {
      printf("%c",A[i]);
    }
  }
```

```
#include<stdio.h>
  char * Day of week(int)
int main()
 {
   int DayNo;
   char day;
   Printf (" Enter Day No ");
   scanf ("%d", &DayNo);
   char *Dayofweek( DayNo);
   If (Day)
    Printf ("%s", * Day of week);
   else
     Printf ("Invalid day ");
  }
   char *Day of week (int day)
  {
    char *weekday [7] ={"S", "M", "T", "W", "TH", "F", "SAT"};
    If ( Day >=1 && Day <=7)
      return (*weekday -1);
    else
      return NULL;
  }
```

Lucky Date Page

Types of recursion

Mainly two type of recursion.

① Direct recursion
② Indirect recursion

① Direct recursion
when a function call itself in a program then it is called direct recursion.

Ex :-

```
void f1();
int main()
{
  // statement;
}
void f2()
{
  // statement;
  f1()
  // statement;
}
```

② Indirect recursion
when a function call another function then it is called Indirect recursion.

```
void f1();
void f2();
int main()
{ statement;
}
```

```
void f1()
{
    // statement;
    f2();
    // statement;
}
void f2()
{
    // statement;
    f1();
    // statement;
}
```

Que

```
#include <stdio.h>
int result(int);
int main()
{
    int n,a=0;
    printf("Enter any no:");
    scanf("%d",&n);
    a= result(n);
    printf("result = %d",a);
    return 0;
}
int result(int n)
{
    if (n!=0)
    {
        return n + result(n-1)
    }
    return n;
}
```

Searching Algorithms in c

① linear search / sequential search

→ we search elements from an array using linear search technique.

→ this technique is generally used to search element in sequencial order.

→

Step in linear search

Step1: create a array with define size.

Step2: select elements to search generally known as key or searching element.

Step3: start searching from index zero (0) from given array.

Step4: If element found then Print ~~Print~~ message that element found at position given.

Step5: If element not found then search next element of given array.

Step6: If element found the Print position of element otherwise. go to step 4 and repeat till element found.

Binary Search

case 1 key == mid element

case 2 key < mid element

case 3 key > mid element

	0	1	2	3	4	5
a[6] =	3	5	12	6	9	8

sort

	0	1	2	3	4	5
a[6] =	3	5	6	8	9	12

If key = 6

case I → key == mid element

If key = 9

case I → key > mid element

So we work on, after $\left[\frac{0+5}{2}\right] = 2$

8	9	12

apply case I → key == mid element.

If key = 3

case III → key < mid element

So we work on, before $\left[\frac{0+5}{2}\right] = 2$

3	5	6

Here key != mid element.

apply case I.

Sorting Algorithm

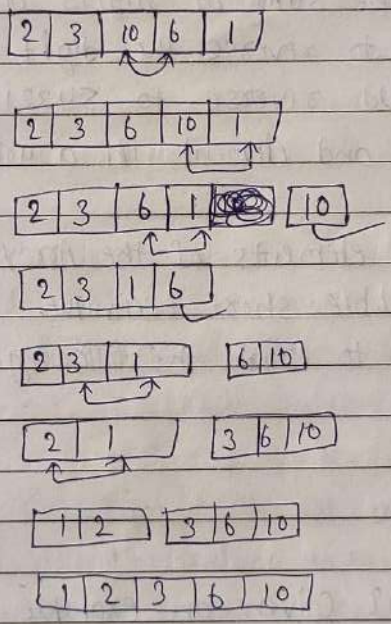
Date
Page

① Bubble sort

	0	1	2	3	4
a[5] =	2	10	3	6	1

pass = 1

```
If (1st > 2nd)
    a[i] > a[i+1]
     swap
else
    continue;
```



File Handling

Lucky Date Page

a=1 → compailer .txt
b=2 → file name . doc
.docx
.xls
.dat

Operation
→ read → "r", "r+" → first read then write
→ write → "w", "w+" → first write then read
→ Append → "a", "a+"

→ include <stdio.h>
→ main();
→ printf();
→ scanf();
→ fgets();
→ fgete();
→ puts();
→ gets();
→ fpute();
→ fopen();
→ fclose();
→ getch(); → <conio.h>

```
#include <stdio.h>
main(){
FILE *p
p = fopen("computer.dat","r");
fprintf("*p","comment");
fclose(*p)
}
```

file path/

read and write operation

```
p = fopen("computer.txt","r+");
fprint("computer.txt");
```

→ Dynamic memory Allocation.

library function <stdlib.h>

default void memory size=0

- malloc(); → memory Allocation
- calloc(); → clear Allocation
- realloc(); → Re-allocation
- free(); → Free-allocation

malloc	calloc
(i) No. of block not define	(i) No. of block define
(ii) white type	(ii) white type
(iii) discontinous memory location	(iii) continous memory location

Lucky

Date ___/___/___

Page _____

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
  int *p, i;
  p= (*int) malloc (sizeof int);
  for(i=0; i<=5; i++)
  {
    printf("%d", p+i*2);
  }
}
```

Typecasting

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
  int *p, n;
  printf("Enter the no. of block");
  scanf("%d", &n);
  p= (*int) calloc(n, (sizeof int));
  for(int i=0; i<=n; i++)
  {
    printf("%d", p+i);
  }
}
```

~~#inc~~

PPS

Lucky Date ___ ___ ___
Page _____

objective.

① A variable name in C should be a/an.
(i) reserve word
(ii) Keyword
(iii) identifier
(iv) All of the above.

② Why do we write (int*) before malloc ?
(i) It is for the syntax correctness
(ii) It is for the type casting.
(iii) It is to inform malloc function about data-type expected.
(iv) None of the above.

③ which header File is used for reading and writting to a file ?
(i) #include <iostream.h>
(ii) #include <fStream.h>
(iii) #include <File.h>
(iv) All of the above.

④ which of the following functions displays the keyboard charactor for Pressed key ?
(i) getch ()
(ii) getche ()
(iii) Both (i) and (ii)
(iv) None of the above.

⑤ what does the following declaration mean ?
int (*ptr) [10];
(i) ptr is array of pointers to 10 integers.
(ii) ptr is a pointer to an array of 10 integers
(iii) ptr is an array of 10 integers.
(iv) ptr is a pointer to array.

⑥ Automatic variables are initialized to.
(i) zero
(ii) Junk value
(iii) None of the above
(iv) Both of the above.

⑦ which of the following symbols is the type-specifier?
(i) &
(ii) *
(iii) #
(iv) %.

⑧ The && and || operators
(i) compare two numeric values.
(ii) compare two numeric values short-hand operators
(iii) compare two Boolean values
(iv) None of the above.

⑨ Identify the correct sequence of steps to run a program.
(i) Link, load, code, compile and execute.
(ii) code, compile, link, execute and load.
(iii) code, compile, link, load and execute.
(iv) compile, code, link, load and execute.

⑩ An external variable is the one
(i) which resides in the memory till the end of of the program.
(ii) which is globally accessible by all the functions.
(iii) which is declared outside the body of the function
(iv) All of the above.

(11) A program attemps to generate as many permutations as possible of the string, 'abcd' by pushing the characters a,b,c,d in the same order onto a stack, but it may pop off the top character at any time. Which one of the following strings cannot be generated using this program?

(i) abcd
(ii) dcba
(iii) cabd
(iv) cbad

(12) Suppose a c program has floating constant 1.414, what is the best way to convert this "float" data type ?

(i) (float) 1.414
(ii) Float (1.414)
(iii) 1.414f or 1.414F
(iv) 1.414 itself of "float" data type, i.e, nothing else required.

(13) user defined data type can be derived by

(i) struct
(ii) enum
(iii) typedef
(iv) All of the above.

(14) As per c language standard, which of the following is/are not keywords) ? Pick the best statement:
auto, make, main, sizeof, elseif

(i) make, main
(ii) sizeof
(iii) auto, make
(iv) make, main, elseif

Lucky Date __/__/____ Page _____

(15) when does the segmentation fault occur?
(i) Compile-time
(ii) Run-time
(iii) Both of the above
(iv) None of the above.

(16) Process of inserting an element in stack is called.
(i) create
(ii) push
(iii) evaluation
(iv) Pop

(17)

	char	int	Float	double
• Format specifier:	%c	%d	%f	%lf
• Range	-128 to 127	-2147483648 to 2147483647	±1.2e-38 to ±3.4e+38	±2.3e-308 to ±1.7e+308

Subjective.

① Discuss about some methods or methodologies used for Problem solving.

→ The computer is the symbol-manipulating machine that follows the set of instructions called a program. any computing has to be performed independently without depending on the Programming language and the computer.

The Problem solving techniques involves the following steps
- Define the Problem.
- Formulate the mathematical model.
- Develop an algorithm.
- write the code for the Problem.
- Test the Program.

Lucky Date ___ ___ ___
Page _____

② How Programming utilizes different components of computer system?

→ Programming utilizes different components of a computer system to perform various tasks and achieve desired functionality. Here are some key components how Programming interacts with them.

1. central Processing Unit (CPU):- The CPU is responsible for executing ~~not~~ instructions and performing calculations. controlling the flow of the Program and utilizing the CPU's capabilities efficiently.
2. memory: memory, including RAM is used to store data and instructions during Program execution.
3. operating system: The operating system provides an interface between the hardware and software components of a computer system. Programming interacts with the operating system through system calls and APIs to perform tasks
4. Input & output devices: Input devices provide data input to processor, which Processes data and generates useful information that's displayed to the user through output devices.

③ what is the role of 'Header files' in C language? List any four with their usage.

→ A header File is a file with extension .h which contains C function declarations and macro definitions to be shared between several source Files.

Lucky Date __/__/__
Page

1. <stdio.h> (standard input/output):- This header file provides function and definitions for standard input and output operations. It includes functions like printf(), scanf(), fopen(), fclose() and constants like NULL, EOF and data type like FILE.
2. <math.h> (mathematics):- This header file contains mathematical function and constant. it includes Function like sqrt(), pow(), sin(), cos(), log() and constants like M_PI (Pi), M_E(Euler's number).
3. <string.h> (String operations):- This header file provides functions for manipulating string and character arrays. It includes function like strlen(), strcpy(), strcat(), strcmp.
4. <stdlib.h> (standard Library):- This header file ~~provides functions~~ contains general-purpose function and memory allocation utilities. it includes function like malloc(), free(), exit() rand()

Q) write the syntax for conditional operator.

→ The conditional operator in c, is also known as the ternary operator.

syntax:-

condition ? expression1 : expression2;

- condition:- This is the condition that evaluates to either true or false. if the condition is true the expression1 is evaluated. otherwise the expression2 is evaluated.

- expression 1 :- This is the value or expression returned if the condition is true.
- expression 2 :- This is the value or expression returned if the condition is false.

(5) what do you mean by Pre Processor? Also explain the use of %i format specifier w.r.t scanf() function.

→ The Pre Processor is a Phase that occurs before the actual compilation Process. It is responsible for performing various tasks such as including header files, macro expanson, conditional compilation. The Preprocessor directives start with a hash symbol (#)

~~Some Common uses of the Preprocessor include:~~

Some example of Pre Processor

#include, #define, #if, #elif

%i format specifier in scanf(), it is used for reading integer value from the input.

(6) what do you mean by 'call by refrence' Explain with the help of a function which swaps two numbers.

→ In the call by refrence method, there is a modification in the original value. It Passes address value.

Example:-

```
#include <stdio.h>
int swap();
int main()
{
  int x=10, y=20;
  swap(&x, &y);
}
int swap(int *ptr1, int *ptr2)
{
   int z = *ptr1;
   *ptr1 = *ptr2;
   *ptr2 = z;
   printf("x=%d, y=%d", *ptr1, *ptr2);
}
```

(7) WAP to Find the number of lines in a text File.

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
   FILE *fp;
   fp = fopen("abc.txt", "r");
   char ch;
   int lines;
   fp = fopen("abc.txt", "r");
   if (fp == NULL)
   {
     printf("Error while opening a file");
     return -1;
   }
```

```
while ((ch=fgetc(fp)) != EOF)
{
    if (ch == '\n')
    {
        lines++;
    }
}
fclose(fp);
printf("Number of lines in a file are :%d", lines);
return 0;
}
```

⑥ WAP to Reverse a string without using strrev().

```
#include<stdio.h>
int main()
{
    int i,l;
    char str[100];
    printf("Enter a string:");
    scanf("%s", str);
    l=0;
    while (str[l] != '\0')
    {
        l++;
    }
    i=l-1;
    while (i>=0)
    {
        printf("%c", str[i]);
        i--;
    }
    return 0;
}
```

(7) concatenate two strings without using strcat().

```
#include<stdio.h>
#include<string.h>
int main()
{
 int len1, len2, i;
 char s1[30], s2[30];
 printf("Enter First string:");
 gets(s1);
 printf("Enter second string:");
 gets(s2);

 len1 = strlen(s1);
 len2 = strlen(s2);
 for(i=0; i<=len2; i++)
 {
    s1[len1+i] = s2[i]
 }
 printf("string After concatenate :%s", s1);
}
```

(8) WAP to print Fibonacci series.

```
#include<stdio.h>
int main()
{
 int a,b,n, sum, i;
 a=0;
 b=1;
```

```
Printf("enter the number of terms : ");
scanf("%d", &n);
for(i=1; i<=n; i++)
{
  Printf("%d",a);
  sum = a+b;
  a = b;
  b = sum;
}
return 0;
}
```

(9) factorial of a given number using recursion.

```
#include<stdio.h>
int main()
{ int fact();
int fact();
int main()
{
  int n;
  Printf("Enter Number:");
  Scanf("%d", &n);
  fact(n);
  Printf("factorial is %d", fact);
}
int fact(int n)
{
  if(n==0)
  return 1;
int factNm1 = fact(n-1);
int factN = factNm1 * n;
return factN;
}
```

Lucky Date __/__/__ Page _____

(10) What are library functions and their uses in C language? Can we write own function and include them in C library.?

→ In C Programming, library functions are Pre-defined functions that are Provided by libraries and can be used by Programmers to perform specific tasks. These functions are usually implemented in standard libraries and offer read to use functionality, saving developers time and effort. library functions are ~~not~~ typically categorized based on their specific purpose, such as string manipulation mathematical operations, input/output, memory management etc.

Here are some example used in library functions

① Printf() and scanf() in <stdio.h>
② strlen, strcpy(), strcat() in <string.h>
③ sqrt, Pow() in <math.h>
④ malloc(), free() in <stdlib.h>

(11) With the help of an example, differenciate between static and dynamic memory allocations

→

Static memory allocation	dynamic memory allocation
① Static memory allocation is done before Program execution.	① Dynamic memory allocation is done during Program execution
② It uses stack for managing the static allocation of memory	② It uses heap for managing the dynamics allocation of memory.
③ It is less efficient	③ It is more efficient.
④ there is no memory re-usability	④ there is memory re-usability

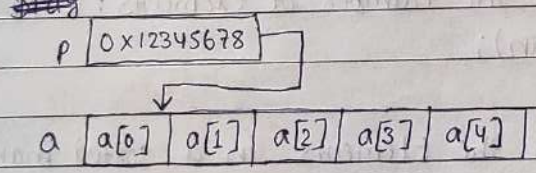
⑤ In this memory is allocated at compile time.	⑤ In this memory is allocated at run time.
⑥ It is generally use for array.	⑥ It is generally use for Linked list.

(12) How array and pointer are related? Explain with the help of suitable diagram.

→ An array is represented by a variable that is associated with the address of its first storage location. A pointer is also the address of a storage location with defined type, so D permits the use of the array [] index notation with both pointer variables and array variables.

Example

```
int a[10];
int *p;
p=a ; //Points to the a[0] location.
```



(13) Types of operators in C

→ (i) Arithmetic operator
(ii) Relational operator
(iii) Logical operator
(iv) Assignment operator
(v) increment & decrement operator.
(vi) Shift operator.

Lucky Date __/__/__ Page _____

(14) What is File handling in C.

→ File handling in C enables us to create, update, read and delete the files stored on the local file system through our C program. The following operations can be performed on a file.

- creation of the new file.
- opening an existing file. fopen()
- Reading from the file. fscanf()
- writing to the file. fprint()
- Deleting the file.

(15) WAP to perform binary search on a set of given sorted number.

```
#include<stdio.h>
int main()
{
 int n,i,a,low,high,mid;
 Printf("Enter the number of elements:\n");
 scanf("%d",&n);
 int arr[n];
 Printf("Enter the Elements in a sorted manner:\n");
 for(i=0;i<n;i++)
 {
  scanf("%d",&arr[i]);
 }
 printf("Enter the value of Find element:");
 scanf("%d",&a);
 low=0;
 high=n-1;
```

```
mid = (low + high)/2;
while(low<=high)
{
    if(arr[mid]<a)
        low = mid+1;
    else if(array[mid]==a)
    {
        printf("%d found at the location %d", a, mid+1);
        break;
    }
    else
    high= mid-1
    mid = (low+high)/2
}
if(low > high)
 printf("%d is not present in the array", a);
return 0;
}
```

//

```
linear search.
for(i=0; i<n; i++)
{
    if(arr[i]==a)
    printf("%d is found at the location of %d", a, i+1);
    return 0;
}
```

(16) WAP to Print Armstrong number.

```c
// WAP to Print Armstrong number
#include<stdio.h>
#include<math.h>
int main()
{
    int n;
    Printf("Enter Number:");
    scanf("%d", &n);
    int z=n;
    int i, x=0;
    for(i=1 ; ; i=10*i)
    {
        if (n/i!=0)
        {
            x++;
        }
        else
        break;
    }
    int sum=0;
    while(n>0)
    {
        int r= n%10;
        int s= Pow(r,x);
        sum= sum+s;
        n= n/10;
    }
    if (sum==z)
    Printf("Armstrong Number");
    else
    Printf("Not Armstrong Number");
    return 0;
}
```

```c
if (n/i!=0)
{
    x++;
}
else
break;
}
int sum=0
while (n>0)
```

$P\left(1+\frac{r}{n}\right)^{nt}$

Lucky Date Page $P\left(1+\frac{r}{n}\right)^{nt}$

(17) simple interest & compound interest.

simple interest: $\frac{Prt}{100}$

compound interest: $P\left(1+\frac{r}{n}\right)^{nt}$

- P → Principle
- r → Interest rate (decimal)
- n → Number of times interest is compounded.
- t → Time in years

(18) WAP to reverse the digit of a given number.
(eg:- 12345 ⟶ 54321)

```
#include <stdio.h>
int main()
{
 int n;
 Printf("Enter number: ");
 scanf("%d", &n);
 int i;
 printf(" Reverse Number is ");
 while(n>0)
 {
    int r = n%10
    Printf("%d", r);
    n= n/10;
 }
 return 0;
}
```

Lucky Date ___/___/___ Page ___

(3) What is algorithm? Given an example algorithm for any problem.

→ An algorithm is a set of instructions for solving a problem.

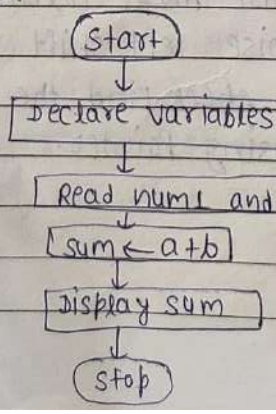
Example: - ① Algorithm code to add two numbers.

step 1 : Start
step 2 : Read the two numbers in to a, b
step 3 : c = a+b
step 4 : write/Print C
step 5 : stop.

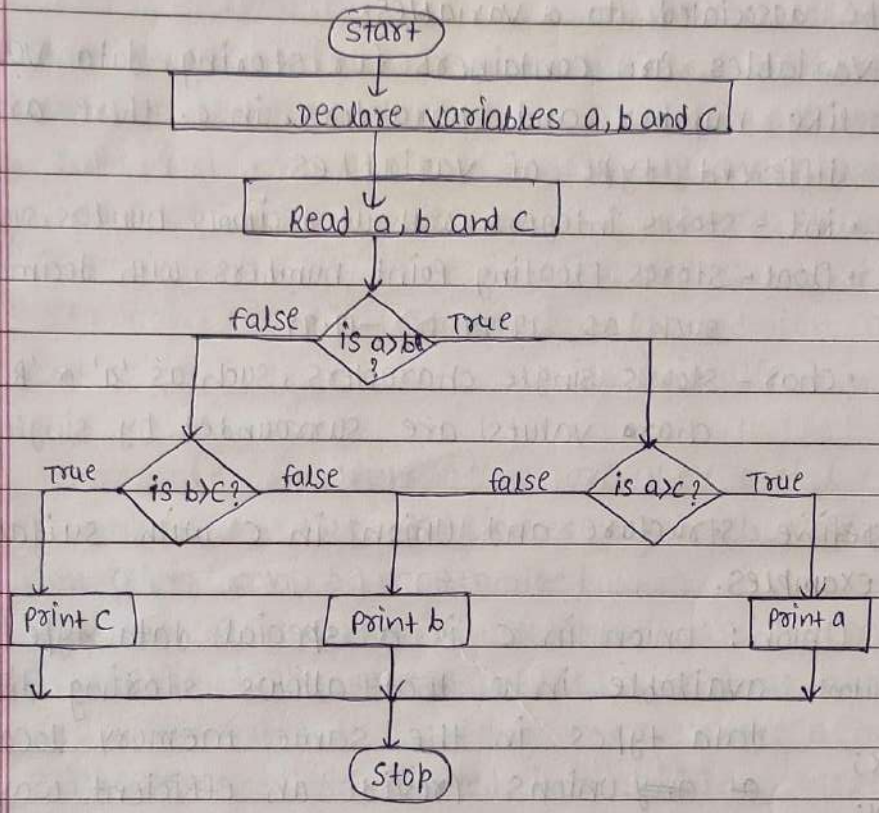
Example ② Algorithm to find out number is odd or even?

step 1 : start
step 2 : input number
step 3 : rem = number mod 2
step 4 : If rem = 0 then
Print "number even"
else
Print "number odd"
end if
step 5: stop.

Flowchart



Flowchart to find the largest among three different numbers entered by user.



(13) Define data types in C.

→ Data type is a set of values with similar characteristics Data type in the C Programming language are used to specify what kind of value can be stored in a variable.

In C language 2 different type of data types.

① Primary data type

Ex:- int, char, float, void

② Derived data type

Ex:- Array, structure, union and pointer.

Variables is the name of memory location which stores some data.

(13) What are variables? List of data types which can be associated to a variables.

→ Variables are containers for storing data values, like number and characters. in c. there are different type of variables

- int - stores integers, without decimals numbers, such as 123
- float - stores floating point numbers, with decimals such as 19.99 or -19.99
- Char - stores single characters, such as 'a' or 'B'. char values are surrounded by single quotes.

(25) Define structure and Union in c with suitable examples.

→ Union: union in c is a special data type available in c that allows storing different data types in the same memory location. Unions provide an efficient way of using the same memory location for multiple purposes. To define a union, we use the union statement.

Ex:-
```
union student
{
  char x;
  float y;
} mark;
```

Structure: Structure in c is a user-defined data type available in c that allows to combining of data items of different kinds. To define a structure you must use the struct statement.

Ex:-
```
struct student
{
  char x;
  float y;
} mark;
```

Lucky Date ___ ___ ___
Page _____

(18) Difference between Structure and Union.

Structure	Union
① The keyword struct is used to define a structure	① The keyword union is used to define a structure.
② Individual member can be accessed at a time.	② only one member can be accessed at a time.
③ Several members of a structure can initialize at once.	③ only the first member of a union can be initialized.

(17) What are function? How are they useful? comment on two categories in which various function can be categorized.

→ function is a group of statements which perform specific task. ~~where~~ It is a modular approach ~~function makes~~ to writing code, where a large program can be divided into smaller, more manageable parts. functions can be called from any part of the ~~function~~ program where they are needed, which makes them very useful for code reuse and makes the program easier to read and maintain.

function in C can be categorized into two broad categories

① library functions: These are built-in functions that are provided by the C standard library. They are included in the program by including the corresponding header file.
Ex:- Printf(), Scanf(), Pow() etc.

② user-defined function: These are functions that are created by the programmer to perform a specific task. Ex:- addNumber(), displayMenu() etc.

Lucky Date ___/___/___ Page ___

(20) Explain pointer ~~air~~ Arithmetic with the help of suitable examples.

→ Pointer arithmetic is performed using the pointer operators (+, -, ++, --), which add or subtract an integer value to or from the address stored in the pointer. ~~the~~

Example:-

```
#include <stdio.h>
int main()
{
    int arr[] = {10, 20, 30, 40, 50};
    int *p = arr; // p points to the first element of arr
    printf("value of *p = %d \n", *p); // prints 10
    printf("value of *(p+1) = %d \n", *(p+1)); // prints 20
    printf("value of *(p+2) = %d \n", *(p+2)); // prints 30

    p++; // p now points to the second element of arr
    printf("value of *p after increement = %d \n", *p);
                                        // Prints 20
    p--; // p now points back to the first element
          of arr.
    printf("value of *p after decrement = %d \n", *p);
                                        // prints 10
    return 0;
}
```

Lucky Date ___ ___ ___
Page _____

(24) What is recursive function in C.

→ Any function that happens to call itself again and again (directly or indirectly), unless the program satisfies some specific condition is called a recursive function.

(14) what is the difference between compilation and execution of a program.

→ Compilation is the process of converting source code into machine code, while execution is the process of running the compiled program. Compilation is typically done once, before the program is executed, while execution can occur multiple times.

(25) what is function deceleration, function definition and function calling. Explain suitable example.

→ Function deceleration: In this part we decelere the function for identification.

Syntax
↳ data_type Name_of_function(parameters);

Ex:-

```
int sum(int x);
```

function definition: In this part we defined the function in details with its working methods.

Syntax
↳

```
data_type Name_of_function(Parameters)
{
  // body of function;
  return;
}
```

Lucky Date __/__/__ Page _____

function calling: ~~In this part~~ It is used to call the defined functions as per requirment.

Syntax:

~~data~~ Name_of_function(argument);

Ex:-

~~sum(int x, int y)~~

sum(x, y);

sum( );

(27) Explain various types of arithmetic and logical operators.

→ Arithmetic operators are used to perform mathematical operations on numerical values, while logical operators are used to perform Boolean logic operations on Boolean values (true or false).

Here are the various types of Arithmetic and logical operators:

1. Arithmetic Operators:
   - Addition (+)
   - subtraction (-)
   - multiplication (*)
   - Division (/)
   - modulo (%)
   - Increment (++)
   - Decrement (--)
2. Logical operators:
   - AND (&&): return true if both operands are true.
   - OR (||): return true if at least one operands is true.
   - NOT (!): return the opposite of the operand truth value (if true, returns false, and vice versa).

Lucky Date ___ ___ ___
Page _____

(28) Explain

(i) Bitwise AND operator

The output of bitwise AND is 1 if the corresponding bits of two operands is 1. If either bit of an operand is 0, the result of corresponding bit is evaluated to 0. The bitwise AND operator is denoted by & (ampersand).

(ii) Bitwise OR operator

The output of bitwise OR is 1 if at least one corresponding bit of two operands is 1. Bitwise OR operator is denoted by |.

(iii) Bitwise XOR operator

The result of bitwise XOR operator is 1 if the corresponding bits of two operands are opposite. It is denoted by ^.

(iv) Bitwise complement operator.

Bitwise complement operator is a unary operator. It changes 1 to 0 and 0 to 1. It is denoted by ~.

(v) Right shift operator.

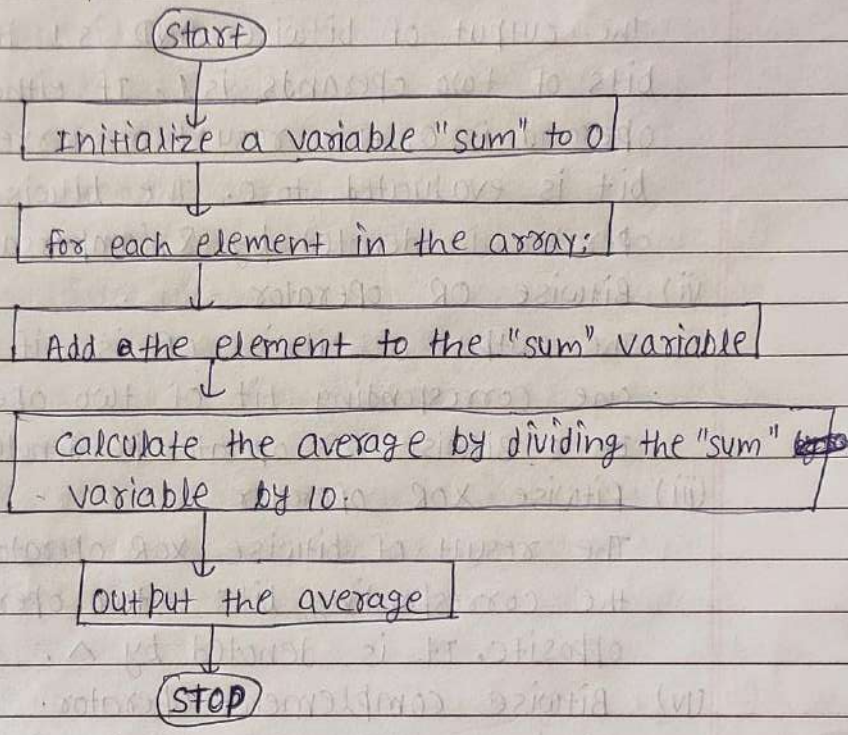
Right shift operator shifts all bits towards right by certain number of specified bits. It is denoted by >>.

(vi) Left shift operator.

Left shift operator shifts all bits towards left by a certain number of specified bits. The bit positions that have been vacated by the left shift operator are filled with 0. It is denoted by <<.

Lucky Date ___/___/___ Page _____

(16) Draw a flowchart to find the average of 10 numbers in an array:



(17)

call by value	call by Reference.
→ In the call by value method, there is no modification in the original value.	→ In the call by Refesence method, there is a modification in the orginal value.
→ It passes original value	→ It passes address value.

(18)