```
Diff b2n Lang. VS program Lang

H2H => Language
H2C => Program Language  C C++
   I
Procedural Language

//32 Keywords
// "Compiler" and Interpreter
// Operators,  Decision, Loop, (iterative) Switch, Function Pointers Array, String FILE ..... [END OF C]

// ..............................SOURCE CODE ..................
#include <stdio.h>

int main()
{
    printf("%f",c);
    return 0;
}
// ..............................

//Processing
 Source Code (.c) => "PREPROCESSOR" =>

// ..............................  EXTENDED SOURCE CODE ..................
int main()
{
    printf("%f",c);
    return 0;
}
// ..............................

```

```c
#include <stdio.h>

int main()
{
    int x, y, z;
    printf("Enter two Numbers");
    scanf("%d %d", &x, &y);
    z=x+y;
    printf("Sum of %d and %d is %d",x,y,z);
    return 0;
}



// 3 basic Data types int (%d) 6  9  2  3     float (%f) 4.5 3.9      Char(%c)   'a' 'b' 'c'
```

input

```
Enter two Numbers4
6
Sum of 4 and 6 is 10

...Program finished with exit code 0
Press ENTER to exit console.
```

```c
#include <stdio.h>

int main()
{
    float x, y, z;
    printf("Enter two Numbers");
    scanf("%f %f", &x, &y);
    z=x+y;
    printf("Sum of %f and %f is %f",x,y,z);
    return 0;
}



// 3 basic Data types int (%d) 6  9  2  3     float (%f) 4.5 3.9      Char(%c)   'a' 'b' 'c'
```

input

```
Enter two Numbers2.5
3.7
Sum of 2.500000 and 3.700000 is 6.200000

...Program finished with exit code 0
Press ENTER to exit console.
```

```
//Processing
Source Code (.C) => "PREPROCESSOR" => Extended Source Code (.S) => "COMPILER" => Assembly Code (.ASM) => "ASSEMBLER"

=> Object File (.obj) of Source Code => "LINKER" => A single file without any extension => "LOADER" => Output File (.exe)


//Source      Program      Process
//Code

//HDD         RAM          PROCESSOR
```

```
// 3 basic Data types int (%d) 6  9  2  3     float (%f) 4.5 3.9     Char(%c)   'a' 'b' 'c'


x Value  2.5  (x)
memory address 1000  (&x)


// Operators
1.  Arithmatic Operators
+ - * / % etc..
    / and % float DType will not work
    -5%2 = -1   but  5%-2 = 1   (Sign depends on the numerator)


2.  Relational Operators will return
    NONZERO or ZERO
    YES or NO
    TRUE or FALSE

    >  >=  <  <=  ==  !=

    int a=3, b=2, c=2;
    d= a>b;
    printf("%d", d);   // OUTPUT 1
```

```
30      TRUE(1) or FALSE(0)
31
32      >  >=  <  <=  ==  !=
33
34      int a=3, b=2, c=2;
35      d= a>b;
36      printf("%d", d);    // OUTPUT 1
37      d= a<b;
38      printf("%d", d);    // OUTPUT 0
39      d= a==b;
40      printf("%d", d);    // OUTPUT 0
41      d= a!=b;
42      printf("%d", d);    // OUTPUT 1
43
44
```

```
3. Assignment Operator
   a=3

4. Logical Operators will return
    NONZERO or ZERO
    YES or NO
    TRUE(1) or FALSE(0)

   &&  Logical AND    (It will return 0 if anyone of the INPUT is 0)
   ||  Logical OR     (It will return 1 if anyone of the INPUT is 1)
   !   Logical NOT    (will reverse the input as output)


   int a=3 (Nonzero), b=-2(Nonzero), c=0 (zero);
    d= a && b && c;      ZERO
    d= a || b || c;      ONE (only considering the value of a)
    d= a || b && c;      ONE (only considering the value of a)
    d= c && a || b;      ZERO
    d= a && c || b;      ONE
```

```
Chapter 3: Decision
Chapter 4: Loops


//swap two numbers
#include <stdio.h>

int main()
{
    int x=13, y=15, t;
    printf("Enter two numbers\t");
 //  scanf("%d%d",&x, &y);

    printf("\nBefore Swap:\t x=%5d and y=%5d",x,y);
//      t=x;      x=y;      y=t;                    // Method 1 : using third variable

    x=x+y;      y=x-y;      x=x-y;    //Method 2 : using + and - operator

//      x=x*y;      y=x/y;      x=x/y;    //Method 3 : using * and / operator
    printf("\nAfter Swap:\t x=%5d and y=%5d",x,y);
    return 0;
}
//*/
```

```c
// Decision:  if (singleton)    if-else (binary)    nested if-else or Switch (multi-decision)

/*
#include <stdio.h>
int main()
{
    int a;
    printf("\nEnter a number\t");
    scanf("%d",&a);

    if(a>10)
     printf("Hi we got a number Greater than 10");

    return 0;
}
*/

```

```c
// even odd
#include <stdio.h>
int main()
{
    int a;
    printf("\nEnter a number\t");
    scanf("%d",&a);

    if(a%2 == 0)
     printf("Even");
    else
      printf("Odd");

    return 0;
}
```

```c

if(2+3*2)          //Conditional Statement/expression will be evaluated first & then check wheather the value is Non0 or 0
  printf("OK");
if (4%2)
  printf("NOT PRINTED");
if(2-2)
  printf("NOT PRINTED");
if (-2)
  printf("PRINTED");

int a=10;
if(a==10)
  printf("PRINTED");

int a=10;
if(a=10)
  printf("PRINTED");

int a=10;
if(a=5)
  printf("PRINTED");

int a=10;
if(a=0)
  printf("NOT PRINTED");

```

```c
62
63  if(-5)
64    printf("PRINTED");
65
66  if()
67    printf("PRINTED");   //ERROR : NO Conditions found
68
69  int a;
70  if(a=0)
71    printf("NOT PRINTED");
72
73  int a=2, b=3;
74  if(a>b)
75    printf("NOT PRINTED");
76
77
78  int a=2, b=3;
79  if(a>b)          // for FALSE expression it will ignore the immediate below line'
80    printf("HELLO");
81  printf("CU");
```

```c
// C is a free-form language.  ; is end of statement/expression.  ; is the delimeter

#include <stdio.h>
int main()
{
    int a;
     printf("\nEnter a number\t");
     scanf("%d",&a);

     if(a>10)
      printf("Hi we got a number Greater than 10");

     return 0;
}
```

```
91
92      int a=5;
93      if(a<1);
94      printf("PRINTED");
95
96      int a=5;
97      if(a<1)
98      printf("NOT PRINTED");
99
100
```

```c
// Nested If Else

int tot_marks;
scanf("%d",&tot_marks);

if(tot_marks>90)
  printf("Grade O");
else if(tot_marks>80)
  printf("Grade A");
else if(tot_marks>70)
  printf("Grade B");
else if(tot_marks>60)
  printf("Grade C");
else                              //default
  printf("Grade F");
```

```
// for more than one statements  {} is must
```

```
125  //Conditional Operator
126
127   if(a>b)
128    c=1;
129   else
130    c=0;
131
132   c = a>b ? 1 : 0;
133   //variable =  condition ? TRUE : FLASE
134
135
```

```c
// even odd
#include <stdio.h>
int main()
{
    int a;
    printf("\nEnter a number\t");
    scanf("%d",&a);

    if(a%2 == 0)     printf("Even");     else       printf("Odd");  // if is a decision Keyword

    a%2 == 0 ?  printf("Even"); : printf("Odd");    // conditional Operator

    return 0;
}
```

```
152
153     printf("Enter a number");
154     scanf("%d%d",&x,&y);
155     max = x>y ? x : y;
156     min = x<y ? x : y;
157     printf("Max is %d and min is %d", max, min);
158
```

```c
#include <stdio.h>

int main()
{
   int i=12,j;

   //printf returns no of characters "printed" by the printf function
   j=printf("%d%d",i, i);
   printf("%3d",j);

   j=printf("%d %d",i, i);
   printf("%3d",j);

   j=printf("%2d %3d",i, i);
   printf("%3d",j);

   int i=123,j;
   j=printf("%2d",i);
   printf("%d",j);

   //Scanf return no of inputs "received" by the scanf function
   j=scanf("%d%d",&i,&i);
   printf("%3d",j);
  return 0;
}
```

```
// Loop   "repetition"
// when we try to "do something more than one times"
#include <stdio.h>
int main()
{
   //Execution Style of FOR loop
       1    2     3
   for(i=0; i<10; i++)     // increment
     printf("print me");

   // 1 2
   // 3 2
   // 3 2
   // 3 2
   // 3  // exit from the loop

   // Output:   0 1 2 3 4 5 6 7 8 9

   //---------------------------
   for(i=0; i<10; i++)      //one statement => No loop braces
     printf("print me");

   //---------------------------
   for(i=0; i<10; i++)     //more than one statement => use loop braces
   {
     printf("print me");
     printf("print me");
   }

   //---------------------------
   for(i=10; i>0; i--)     // decrement
     printf("print me");

  //---------------------------
   for(i=0; i<10; i++)     // increment
     printf("print me");
```

```
   // for(initialization; checking/decision;  increment/decrement)
  //---------------------------
   i=0;                // initialization "outside"
   for(; i<10; i++)
     printf("print me");

   //---------------------------
   for(i=0; ;i++)
   {
      if(i<10)         // decisions "inside Loop"
      printf("print me");
   }

   //---------------------------
   for(i=0; i<10;)
   {
      printf("print me");
      i++;             // increment/decrement "inside Loop"
   }

   //---------------------------
   i=0;                // initialization "outside"
   for(; ;)            // this for loop is valid too
   {
      if(i<10)         // decisions "inside Loop"
        printf("print me");

      i++;             // increment/decrement "inside Loop"
   }



   return 0;
}
```

```
// Loop   "repetition"
// when we try to "do something more than one times"
#include <stdio.h>
int main()
{
   //Execution Style of FOR loop
       1    2     3
   for(i=0; i<10; i++)     // increment
     printf("print me");

   // 1 2
   // 3 2
   // 3 2
   // 3 2
   // 3  // exit from the loop

   // Output:   0 1 2 3 4 5 6 7 8 9

   //---------------------------
   for(i=0; i<10; i++)      //one statement => No loop braces
     printf("print me");

   //---------------------------
   for(i=0; i<10; i++)     //more than one statement => use loop braces
   {
     printf("print me");
     printf("print me");
   }

   //---------------------------
   for(i=10; i>0; i--)     // decrement
     printf("print me");

  //---------------------------
   for(i=0; i<10; i++)     // increment
     printf("print me");
```

```
   // for(initialization; checking/decision;  increment/decrement)
  //---------------------------
   i=0;                // initialization "outside"
   for(; i<10; i++)
     printf("print me");

   //---------------------------
   for(i=0; ;i++)
   {
      if(i<10)         // decisions "inside Loop"
      printf("print me");
   }

   //---------------------------
   for(i=0; i<10;)
   {
      printf("print me");
      i++;             // increment/decrement "inside Loop"
   }

   //---------------------------
   i=0;                // initialization "outside"
   for(; ;)            // this for loop is valid too
   {
      if(i<10)         // decisions "inside Loop"
        printf("print me");

      i++;             // increment/decrement "inside Loop"
   }



   return 0;
}
```

```
// WHILE LOOP

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

   //--------------------------
   i=0;              // Basic WHILE Loop: Increment
   while(i<10)
   {
      printf("print me");
      i++;
   }

   //--------------------------
   i=0;              //  WHILE Loop having ; => Outputs Nothing
   while(i<10);
   {
      printf("print me");
      i++;
   }

   //--------------------------
   i=10;             //Basic WHILE Loop: Decrement
   while(i>0)
   {
      printf("print me");
      i--;
   }

  //--------------------------
   i=8;             //Basic WHILE Loop with Break Statement
   while()          //def. of Break: exits from the current loop or go to the } (closing brace) of nearest loop
   {
      if(i>10)
```

```
        break;
      printf("print me");
      i++;
   }


  //---------------------------
   i=0;            // "nested" WHILE Loop without Break Statement
   while(i<10)
   {
      j=0;
      while(j<5)
      {
         k=0;
         while(k<3)
         {
            printf("print K");
            k++;
         }
         j++;
      }
      i++;
   }

  // How many times?    10 times  * 5 times * 3 times = 150 times "PRINT K"

   //---------------------------
   i=0;            // WHILE Loop with Break Statement
   while(i<10)
   {
      j=0;
      while(j<5)
      {
         k=0;
         while(k<3)
```

```
        {
            if(k==0) //def. of Break: exits from the current loop or just after the } (closing brace) of nearest loop
                break;
            printf("print K");
            k++;
        }
        j++;
    }
    i++;
}

//---------------------------
int i=0;                // Find the Output...
while(i<=32767)
{
    printf("%d",i);
    i++;
}

// Output:
//    0 1 2 3 4 5  .... 32766  32767   ++
//   -32768 -32767 -32766 .... 0 .... 32767
//  -32768 .... 0 ... 32767
//  and so on ....

// Explanation ??   WHY an interger supports any number within a limit of -32768  to +32767
//    an integer variable (for a 32 bit compiler) takes 2B (B bytes b bits)
// 2B = 16b (b bits => 'bi'nary digi't')   => 2^16 combinations are possible
//  65536 combinations =>   32768 nos are used for +ve and 32768 nos are used for -ve
//  so the range will be -32768 ... 0 ... +32767 (range of an int variable)
```

//Character Variable : 1B (32 bit compiler) 2B (64bit compiler)
// 8b => 2^8 combinations => 256 combinations => -128 to +127 is the range of a cahracter variable

// Float ? 4B => 32 bit => 2^32 combinations => HOME TASK

//----------
// PRE INCREMENT / POST INCREMENT OPERATOR

```
int a=2;
a=a+1;  or     a++;  or     a+=1;

int a=2;
a=a-1;  or     a--;  or     a-=1;

int a=2;
a=a*1;  or     a*=1;    //  cant write this one like a**;

int a=2;
a=a/1;  or     a/=1;    //  cant write this one like a//;
```

//---------- Basic Use of

```
                          a  b  c
int a=2, b, c;                2  ?  ?
b= a++;   //post increment      3  2  ?
c= ++a;   //pre increment       4  2  4

                          a  b  c
int a=2, b, c;                2  ?  ?
b= a++;   //post increment      3  2  ?
c= ++b;   //pre increment       3  3  3

                          a  b  c
```

```
    int a=2, b, c;                  2  ?  ?
    b= ++a;   //pre increment         3  3  ?
    c= a++;   //post increment        4  3  3

                                a  b  c
    int a=2, b, c;                  2  ?  ?
    b= ++a * ++a;                     4  16 ?
    b= a++ * a++;                     6  16 ?

                                a  b  c
    int a=2, b, c;                  2  ?  ?
    b= ++a * a++;                     4  9  ?

    //---------------------------
    i=0;              // Pre Increment operator
    while(++i<=10)
      printf("%d",i);     Output?  1 2 3 ...

    //---------------------------
    i=0;              // Post Increment operator
    while(i++<10)
      printf("%d",i);    Output?  1 2 3 ...

    //---------------------------
    i=10;              // Post Increment operator: Shortest Condition
    while(i--)
      printf("%d",i);    //Find the Output ?   9 ... 0 will be the output

    return 0;
}
```

```
#include <stdio.h>
int main()
{
   //--------------------------- Using Break part 1
   for(i=0; i<10; i++)
   {
      if(i>5)
        break;
      printf("%2d",i);       // output: 0 1 2 3 4 5
   }

   //--------------------------- Using Break part 2
   for(i=0; i<10; i++)
   {
      printf("%2d",i);       // output: 0 1 2 3 4 5
      if(i>5)
        break;
   }

   //--------------------------- Using Continue Keyword
   // def. of Break:    exits from the current loop or go to the } (closing
brace) of nearest loop
   // def. of Continue: Starts with the current loop or go to the { (Opening
brace) of nearest loop



   //--------------------------- DIFFERENCE B2N BREAK AND CONTINUE
   for(i=0; i<10; i++)
   {
      if(i>=5)
        break;             // exits from line no 33 again if the value < 5
      printf("%2d",i);       // output:  0 1 2 3 4 (using break)
                         // (for the values starting from 5 to 10 the loop "never
runs")
   }
```

```
    //---------------------------
    for(i=0; i<10; i++)
    {
       if(i>=5)
         continue;              // starts from line no 38 again if the value < 5
       printf("%2d",i);         // output:  0 1 2 3 4
    }                           // (for the values starting from 5 to 10 the "loop runs
but no values will get printed")



    //--------------------------- Using Continue part 2
     for(i=0; i<10; i++)
    {
       if(i==5)
         continue;              // starts from line no 101 again if the value < 5
       printf("%2d",i);         // output:  0 1 2 3 4 6 7 8 9
    }

    //--------------------------- Using Break
     for(i=0; i<10; i++)
    {
       if(i==5)
         break;
       printf("%2d",i);         // output:  0 1 2 3 4
    }
    return 0;
}
```

```
#include <stdio.h>
int main()
{
   int i, j, k, n;          // Consider i for each line, j for each Star and k for each
Space
   printf("Enter no of rows:\t");
   scanf("%d",&n);

   for(i=1;i<=n;i++)    // for each line .. ok but how many lines are there? n
lines
   {
      for(k=1;k<= ? ;k++) // will print spaces at each line
         printf(" ");

      for(j=1;j<= ?? ;j++) // will print * at each line
         printf("*");

      printf("\n");
   }
   return 0;
}
```

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1		n	i		FORMULA			n	n-i+1	i-1	FORMULA			n	2*i-1	n-i	FORMULA
2	n	i Row	j Star	k space	T1 Pattern Looks Like		n	i Rows	j Star	k space	T2 Pattern		n	i Rows	j Star	k space	T2 Pattern Looks Like
3	4	1	1		*		4	1	4	0	****		4	1	1	3	*
4		2	2		**			2	3	1	***			2	3	2	***
5		3	3		***			3	2	2	**			3	5	1	*****
6		4	4		****			4	1	3	*			4	7	0	*******
7			i	n-i	FORMULA				n-i+1		FORMULA				2*(n-i)+1	i-1	FORMULA
8	n	i Rows	j Star	k space	T2 Pattern Looks Like		n	i Rows	j Star	k space	T1 Pattern Looks Like		n	i Rows	j Star	k space	T2 Pattern Looks Like
9	4	1	1	3	*		4	1	4		****		4	1	7	0	*******
10		2	2	2	**			2	3		***			2	5	1	*****
11		3	3	1	***			3	2		**			3	3	2	***
12		4	4	0	****			4	1		*			4	1	3	*

```
//HCF AND LCM

//x= 4 and y=16
// Factors            4= 1 2 4   and 16= 1 2 4 8 16
// Common factors        1 2 4
// HCF                    4
// HCF of 2, 3    HCF will be 1


// Multiples     4=  4 8 12 16 20 24 28  32 ...  and  16= 16 32 ....
// Common multiple   16 32...
// LCM of 4 and 16 will be 16


// LCM should exist in between of max(x,y) to x*y     16 to 64
// 2 and 3    LCM 3 to 6
// 2 and 3    HCF 2 to 1
// mn= min(x,y)       e.g. 4
// every HCF should exist within 1 to minimum of the given two numbers

//
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~`
mn= x>y ? y : x;  // mn will contain the minimum of two numbers  for HCF
for(i=mn; i>=1; i--)        //searching for the HCF between two numbers
{
   if(x%i==0   &&  y%i==0)
   {
      printf("HCF will be %d", i);
      break;
   }
}
//
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~`
mx= x>y ? x : y;  // mx will contain the maximum of two numbers  for LCM
```

```
for(i=mx; i<=x*y; i++)        //searching for the LCM between two numbers
{
   if(i%x==0   &&  i%y==0)
   {
      printf("LCM will be %d", i);
      break;
   }
}
//
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~`



// finding out the Sum-of-digits
// input 348    => 3+4+8    output  15  (sum of each digits)

   s=0;
   scanf("%d",&n);
   while(n!=0)
   {
      r= n%10;
      n=n/10;
      s=s+r;
   }
   printf("sum of digit is %d", s);
//
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~`

// reverse a digit   i/p 348   o/p 843

   s=0;
   scanf("%d",&n);
   while(n!=0)
   {
      r= n%10;
```

```
        n=n/10;
        s=s*10+r;
    }
    printf("sum of digit is %d", s);
//
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~`
```

```
// do-while loop


/* ------------------------------------  Qu 1: Diff b2n do-while and while/for
  i=1;   // init...
  do{
   printf("Calcutta University\n");
   i++;
  }while(i>2);

  // ------------------------------------
  i=1;
  while(i>2)
  {
     printf("Calcutta University\n");
     i++;
  }
 */


 // ------------------------------------   Qu 2: applications of Do-While Loop
#include <stdio.h>
int main()
{
   int i, n;          // Consider i for each line, j for each Star and k for each
Space

   do{
     printf("Enter a positive number:\t");
     scanf("%d",&n);   // let n=-5     -19     18
   }while(n<0);

   printf("We have recvd a +ve no %d", n);

   return 0;
}
```

```
#include <stdio.h>
void hello()          // function definition : function body
{
   printf("HI How are you");
}

int main()
{
   hello();    // function calling: calls the hello funtion    //hello is a function
bcz it considers ()      while()   xyz()   if()   ...;
   return 0;
}
```

```
#include <stdio.h>
void hello();   // function prototype:
int main()
{
   hello();    // function calling: calls the hello funtion    //hello is a function
bcz it considers ()      while()   xyz()   if()   ...;
   return 0;
}

void hello()          // function definition : function body
{
   printf("HI How are you");
}
```

```
/*pointer*/

#include <stdio.h>
int main()
{
   int a=2, *p; // p is a pointer that can hold address of a variable
   p=&a;
   printf("value of a is %d", a);   //2
   printf("value of a is %d", *p);  //2
   printf("address of a is %u", &a);   //100
   printf("address of p is %u", &p);    //200



   int a=2, *p, **q, ***r, ****s;   // p, q, r, s are dangling pointers
   p=&a;
   q=&p;
   r=&q;
   s=&r;

  //  variable [value] memory address
  //       a  [2]      100
  //      p  [100]     200
  //      q  [200]     300
  //      r  [300]     400
  //      s  [400]     500

  a  2   &a 100
  p 100  &p 200    *p 2
  q  200 &q 300    *q 100  **q 2
  r  300 &r 300    *r 200  **r 100   ***r 2
  s  400 &s 500    *s 400  **s 300   ***s 200  ****s 2

  //array with pointer
  //note "array name" denotes the "address of its first cell"

  int a[30]={1,2,3,4,5}, *pa;
```

```
  pa=&a[0];  //pa is a pointer that points to an array variable   "pa=a;"

  for(i=0;i<5;i++)
  {
     printf("%d", a[i]);
     printf("%d", *(pa+i));
  }

  e.g.:   0  1    2    3    4   array index
          100  102   104   106  108 array memory address
       a  1   2    3    4    5   array values
       pa 100

       pa+0  100   *(pa+0)  1
       pa+1  102   *(pa+1)  2
       ...
       pa+4  108   *(pa+4)  5

    return 0;
}
```

```
// adding 2 numbers

#include <stdio.h>
int main()
{
   int a,b,c;
   scanf("%d %d",&a, &b);
   c=a+b;
   printf("Sum is %5d\n",c);
   return 0;
}

// adding 2 numbers  using 1000 times (Looping)

#include <stdio.h>
int main()
{
   int a,b,c;
   for(i=0; i<100; i++)
   {
     scanf("%d %d",&a, &b);
     c=a+b;
     printf("Sum is %5d\n",c);
   }
   return 0;
}



// adding 2 numbers  using functions without arguments
#include <stdio.h>

void add()              //function definition is done:
{
   int a, b, c;
   scanf("%d %d",&a, &b);
   c=a+b;
```

```
    printf("Sum is %5d\n",c);
}

int main()
{
    add();    //function calling
    return 0;
}



// adding 2 numbers  using functions with arguments
#include <stdio.h>

void add(int x, int y)              //function definition is done:
{
    c=x+y;
    printf("Sum is %5d\n",c);
}

int main()
{
    int a,b;
    scanf("%d %d",&a, &b);
    add(a, b);    //function calling using arguments but without any return
value
    return 0;
}



// adding 2 numbers  using functions with arguments and return value
#include <stdio.h>

int add(int x, int y)               //function definition is done: x& y are called
formal arguments
{
    return(x+y);
```

```
}

int main()
{
    int a,b;
    scanf("%d %d",&a, &b);
    c= add(a, b);          //function calling using "actual" arguments   //c=5
    printf("Sum is %5d\n",c);
    return 0;
}
```

```
/* ARRAY [definition, operations (insert and delete, traverse)
// Applications (searhing sorting)
// relation with pointers  => string (character array) */

//define array   int a=5;  2Bytes [-32768 to +32767]

// int a=5, b=6, c=7
/*
  variable name  value    address (let)
    a          5       100
    b          6        58
    c          7       250
*/
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/* int a[5];               // a var =>  a[] indicates it is an array    => a[5] a is an
array with 5 cells

index      0   1   2   3   4
value      ?   ?   ?   ?   ?
m. addr     10  12  14  16  18

float a[5];

index      0   1   2   3   4
value      ?   ?   ?   ?   ?
m. addr     20  24  28  32  36

int a[10] = {23,34,56,78,12};
*/



#include <stdio.h>
int main()
{
   int a[100], n, i, data, pos;
```

```
  do{
  printf("\nEnter the no of inputs (within 0 to 99)");
  scanf("%d",&n);          // n will b d array size   n=5
  }while(n<0 || n>99);   //

  for(i=0;i<n;i++)                    // scanning inputs
  {
     printf("\nenter the value for a[%d]",i);
     scanf("%d",&a[i]);
  }

  //_____________________________________________________
  printf("\nArray looks like below...");     // printing array values
  for(i=0;i<n;i++)
    printf("%5d\t", a[i]);

   //_____________________________________________________
   // Now we will insert some element to an array....
   // Inserting an element into an array

   //scanning an element
   printf("\nEnter the data to be insert... ");
   scanf("%d",&data);

   //scanning for position
   do{
      printf("\nEnter the position... ");
      scanf("%d",&pos);
   }while(pos<0 || pos>n);       // array size 6 (0-5) <0      // }while(pos<0 ||
pos>n);

   // 0 1 2 3 4 5 6    // array indices [positions]
   // 1 2 3 4 5 6    // array size 6 n=6    pos=-1 ?? NO   pos<n   // pos 2  and
data 9  (let)
   // 1 2 3 3 4 5 6  (insertion done)
```

```
    i=n;   // size of an array
    while(i>pos)                    // SHIFTER
    {
        a[i]=a[i-1];        // a[6]= a[5]   a[5]= a[4]  a[4] =a[3]   a[i] = a[i-1]
 CLEAR ????
        i--;
    }                       // 1 2 3 3 4 5 6  (insertion not yet)

    a[pos] = data;     // insert the element into that position
                           // 1 2 9 3 4 5 6  (insertion DONE)
//_____________________________________________________
   printf("\nArray looks like below...");     // printing array values
   for(i=0;i<n+1;i++)
      printf("%5d\t", a[i]);
   return 0;
}
```

ARRAY DELETION

/* ARRAY [definition, operations (insert and delete, traverse)

// Applications (searching sorting)

// relation with pointers => string (character array) */

//define array   int a=5; 2Bytes [-32768 to +32767]

// int a=5, b=6, c=7
/*
  variable name value   address (let)
    a   5   100
    b   6   58
    c   7   250
*/
//~~~~~~
/* int a[5];   // a var => a[] indicates it is an array   => a[5] a is an array with 5 cells

index   0   1   2   3   4
value   ?   ?   ?   ?   ?
m. addr   10 12 14 16 18

float a[5];

index   0   1   2   3   4
value   ?   ?   ?   ?   ?
m. addr   20 24 28 32 36

int a[10] = {23,34,56,78,12};
*/

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

  int a[100]={23, 34, 56, 78}, n=4, i, pos=1;

/*   do{


  printf("\nEnter the no of inputs (within 0 to 99)");
  scanf("%d",&n);          // n will b d array size   n=5
  }while(n<0 || n>99);   //


  for(i=0;i<n;i++)                    // scanning inputs
  {
     printf("\nenter the value for a[%d]",i);
     scanf("%d",&a[i]);
  }

*/
  //_____________________________________________________
  printf("\nArray looks like below...");     // printing array values (traverse)


  for(i=0;i<n;i++)
    printf("%5d\t", a[i]);

   //_____________________________________________________

   // Delete an element from some given position

   //scanning for position
 /*
```

```
    do{
        printf("\nEnter the position... ");
        scanf("%d",&pos);
    }while(pos<0 || pos>n);       // array size 6 (0-5) <0      // }while(pos<0 ||
pos>n);

*/

    // 0    1   2   3     index values
    // 23, 34, 56, 78     array elements    // pos =1    n=4

    data= a[pos];

    i=pos;
    while(i<n-1)                    // LEFT SHIFTER
    {
        a[i]=a[i+1];        // a[1] = a[2]    a[2]= a[3]  a[3]=a[4] ....
        i++;
    }
    n--;     //array size will be reduced by 1

//_____________________________________________________
   printf("\nArray looks like below...");     // printing array values
   for(i=0;i<n;i++)
      printf("%5d\t", a[i]);

   return 0;
}
```

LINEAR SEARCHING

```
// linear searching => modified linear searching

#include <stdio.h>
int main()
{
  int a[100]={23, 34, 56, 78, 112}, n=5, i, data= 56;


/*   do{
  printf("\nEnter the no of inputs (within 0 to 99)");
  scanf("%d",&n);          // n will b d array size   n=5
  }while(n<0 || n>99);   //


  for(i=0;i<n;i++)                    // scanning inputs
  {
     printf("\nenter the value for a[%d]",i);
     scanf("%d",&a[i]);
  }
 */
  //_____________________________________________________
  printf("\nArray looks like below...");     // printing array values (traverse)
  for(i=0;i<n;i++)
    printf("%5d\t", a[i]);


   //_____________________________________________  LINEAR
SEARCHING
  for(i=0;i<n;i++)
  {
    if(a[i] == data)
      printf("\n\n%5d is found at %3dth Location\t", a[i], i);
```

```
    }
    return 0;
}

// problems of linear searching:  2 times checking
```

MODIFIED LINEAR SEARCHING

```
// Modified linear searching

#include <stdio.h>
int main()
{
  int a[100]={0, 23, 34, 56, 78, 112}, n=5, i, data= 300;


/*   do{
  printf("\nEnter the no of inputs (within 0 to 99)");
  scanf("%d",&n);          // n will b d array size   n=5
  }while(n<0 || n>99);   //


  for(i=0;i<n;i++)                    // scanning inputs
  {
     printf("\nenter the value for a[%d]",i);
     scanf("%d",&a[i]);
  }
 */

    // Modified linear SEARCHING
    a[0]= data;
    i= n;
    while(a[i]!=data)
       i--;


   printf("\n\n%5d is found at %3dth Location\t", a[i], i);
  return 0;
}
```

```
// index   0   1   2   3   4   5
// array   300, 23, 34, 56, 78, 112              i=5
```

```
//  Array Fundamentals

#include <stdio.h>
int main()
{
    int a[10]={12,23,34,56,67,92}; //array declaration with
initialization
    int n=6; // Size of an array   (no of elements)
    int i, *p;

    // pointer points to an array
    p=a;    //a array name  => first cell address


    for(i=0;i<n;i++)
       printf("%5d", a[i]);  //print array values
    printf("\n");

    for(i=0;i<n;i++)
       printf("%5d", *(a+i));     // a[i] = *(a+i) = *(i+a) =  i[a]
    printf("\n");

    for(i=0;i<n;i++)
       printf("%5d", *(i+a));
    printf("\n");

    for(i=0;i<n;i++)
       printf("%5d", i[a]);
    printf("\n");

  // using pointer variable
    for(i=0;i<n;i++)
    {
        printf("%5d", *p);     // printing values
        p++;            // pointing to next value
    }
    printf("\n");

    i=0;    // using pointer var 2nd e.g.
    p=a;   //pointing to base address of an array
    while(i<n)
    {
        printf("%5d", *(p+i));     // printing values
        i++;
    }
    printf("\n");
    return 0;
}
```

```
// ARRAY Insertion, Deletion, Traverse, Linear Search and MOdified Linear
Search [DONE]
// Array with Pointer

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
// Traversing Array Elements || Without Functions

#include<stdio.h>
int main()
{
    int a[10]={12,23,34,56,67,92};
    int *p, *q;
    p=&a[1];
    q=&a[4];

    printf("%d\t", *q-*p);   //outputs a value of 67-23
    printf("%ld", q-p);   //output:  3 cells holds an interger data
}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
// Traversing Array Elements || Without Functions

#include <stdio.h>
int main()
{
    int marks[5]={1,2,3,4,5}, i;
    printf("Array Values are as follows\t");
    for(i=0;i<5;i++)
       printf("%4d",i);
    return 0;
}
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
// Traversing Array Elements using function || with Functions

#include <stdio.h>
void display();
int main()
{
    int marks[5]={1,2,3,4,5}, i;
    display();
    return 0;
}
void display()
{
    int i;
    printf("Array Values are as follows\t");
    for(i=0;i<5;i++)
       printf("%4d",i);
}
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
// Passing Array Elements to a function || Call by Value

#include <stdio.h>
void display(int);   // function prototype
```

```
int main()
{
    int marks[5]={1,2,3,4,5}, i;
    printf("Array Values are as follows\t");
    for(i=0;i<5;i++)
       display(marks[i]);
    return 0;
}
void display(int x)   // function definition
{
    printf("%4d",x);
}
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
// Passing Array Elements to a function || Call by Reference

#include <stdio.h>
void display(int *);     // function prototype
int main()
{
    int marks[5]={1,2,3,4,5}, i;
    printf("Array Values are as follows\t");
    for(i=0;i<5;i++)
       display(&marks[i]);
    return 0;
}
void display(int *vi)  // function definition
{
    printf("%4d",*vi);
}
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
// Passing Array Elements to a function || Call by Reference || Example 2

#include <stdio.h>
void display(int *);     // function prototype
void show(int **);
int main()
{
    int marks[5]={1,2,3,4,5}, i;
    printf("Array Values are as follows\t");
    for(i=0;i<5;i++)
       display(&marks[i]);
    return 0;
}
void display(int *vi)  // function definition
{
    show(&vi);
}

void show(int **p)
{
    printf("%4d",**p);
}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
// Passing Array Elements to a function || Call by Reference || Example 3

#include <stdio.h>
void display(int *);     // function prototype
void show(int **);
void disp(int ***);
int main()
{
    int marks[5]={1,2,3,4,5}, i;
    printf("Array Values are as follows\t");
    for(i=0;i<5;i++)
       display(&marks[i]);
    return 0;
}
void display(int *vi)  // function definition
{
    show(&vi);
}
void show(int **p)
{
    disp(&p);
}
void disp(int ***t)
{
    printf("%4d",***t);
}
```

```
// Two Dimensional Array Glimpse

#include <stdio.h>
int main()
{
    int a[10][10]={{12,23},{34,56},{67,92}}; //2D array declaration with
initialization
    int r=3, c=2, i, j;
    for(i=0;i<r;i++)                    // for each row
    {
        for(j=0;j<c;j++)                        // for each column
           printf("%5d",a[i][j]);                   // printing each
elements
        printf("\n");
    }
    return 0;
}
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
// 2D Array Initialization
int a[10][10]={{12,23},{34,56},{67,92}};   // Acceptable
int a[][10]={{12,23},{34,56},{67,92}};     // Acceptable
int a[10][]={{12,23},{34,56},{67,92}};    //  ERROR: 2nd Dimension is
very crucial
int a[][]={{12,23},{34,56},{67,92}};      // Acceptable or Not ???

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
// 2D Array Initialization
#include <stdio.h>
int main()
{
    int a[10][10]={{12,23},{34,56},{67,92}}; //2D array declaration with
initialization
    int r=3, c=2, i, j;

    printf("Normal Print\n");
    for(i=0;i<r;i++)                    // for each row
    {
        for(j=0;j<c;j++)                        // for each column
           printf("%5d",a[i][j]);                   // printing each
elements
        printf("\n");
    }

    printf("Print Data using one Dimensional Pointer\n");
    for(i=0;i<r;i++)                    // for each row
    {
        for(j=0;j<c;j++)                        // for each column
           printf("%5d",*(a[i]+j));                  // printing each
elements
        printf("\n");
    }

    printf("Print Data using Double Dimensional Pointer\n");
    for(i=0;i<r;i++)                    // for each row
    {
        for(j=0;j<c;j++)                        // for each column
           printf("%5d",*(*(a+i)+j));                  // printing each
elements
        printf("\n");
    }
    return 0;
}
```

```
// 2D Array Initialization
#include <stdio.h>
// Getting each elements ||  *(Base Address * Row Number * No of Columns
+ Column Number)

void display2d(int *a, int r, int c)
{
    int i, j;
    for(i=0;i<r;i++)                    // for each row
    {
        for(j=0;j<c;j++)                        // for each column
           printf("%5d",*(a+i*c+j));          // printing each elements
        printf("\n");
    }
}

void show2d(int (*a)[2], int r, int c)        // Column value is
important
{
    int i, j, *p;
    for(i=0;i<r;i++)
    {
        p=a+i;                              // pointing each row
        for(j=0;j<c;j++)                    // accessing each column
           printf("%5d",*(p+j));
        printf("\n");
    }
}

void print2d(int a[][2], int r, int c)        // Column value is
important
{
    int i, j;
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
           printf("%5d",a[i][j]);
        printf("\n");
    }
}

int main()
{
    int a[10][10]={12,23,34,56,67,92}; //2D array declaration with
initialization
    int r=3, c=2, i, j;

    printf("Print Data using function | Call by reference\n");
    display2d(a,r,c);

    printf("Print Data using function | Call by reference\n");
    show2d(a,r,c);

    printf("Print Data using function | Call by reference\n");
```

```
    print2d(a,r,c);
    return 0;
}
```

```
// Array insert and delete

#include <stdio.h>
int main()
{
   int a[100], n, i, data, pos;
   printf("\nEnter the no of inputs");
   scanf("%d",&n);

   for(i=0;i<n;i++)                          // input of an array
   {
       printf("\nenter the value for a[%d]",i);
       scanf("%d",&a[i]);
   }
//______
   printf("\nArray looks like below...");     // output of an array
   for(i=0;i<n;i++)
      printf("%5d\t", a[i]);

//______
    // Now we will insert some element to an array        PORE MUKHE
BOLCHI EKHN CODE TA JUST DEKHTE THAK OK ??
    // Inserting an element into an array

    //scanning an element
    printf("\nEnter the data to be insert... ");
    scanf("%d",&data);

    //scanning for position
    do{
      printf("\nEnter the position (for insertion) [range 0 to %d] ...
",n-1);
      scanf("%d",&pos);
    }while(pos<0 || pos>n);


    // 0 1 2 3 4 5 6 7    // positions
    // 1 2 3 9 7 4 5 6       array size 7 n=7    pos=11 ?? NO   pos<n
pos ???
    i=n;   // size of an array
    while(i>pos)                       // SHIFTER
    {
        a[i]=a[i-1];
        i--;
    }
    a[pos] = data;     // insert the element into that position
    n++;     // no of data in that array will be increased ....

//______
   printf("\nAfter Insertion, Array looks like below...");     // output
of an array
   for(i=0;i<n;i++)
      printf("%5d\t", a[i]);
```

```
//_____________________ARRAY DELETION
    // 0 1 2 3 4 5 6 7    // positions
    // 1 2 3 9 7 4 5 6    array size 8 n=8    pos 3

    //scanning for position
    do{
      printf("\nEnter the position (for deletion) [range 0 to %d] ...
",n-1);
      scanf("%d",&pos);
    }while(pos<0 || pos>n);     //pos=11 ?? NO   pos<n   pos ???

    data=a[pos];
    printf("\nThis data %5d is deleted now...", data);

    i=pos;
    while(i<n-1)         //left shifting
    {
       a[i]= a[i+1];
       i++;
    }
    n--;     // no of data in that array will be decreased ....
//______
   printf("\nAfter Deletion, Array looks like below...");     // output
of an array
   for(i=0;i<n;i++)
      printf("%5d\t", a[i]);

   return 0;
}
```

```
//matrix addition (r1 = r2  and c1= c2)

#include <stdio.h>
int main()
{                              //matrix (2D array) declaration with
initialization
    int mat1[10][10]={{12,34,45,24},{56,67,78,39},{42,46,70,30}}, m1r=3,
m1c=4, i, j;
    int mat2[10][10]={{12,34,45,24},{56,67,78,39},{42,46,70,30}}, m2r=3,
m2c=4, mat3[10][10];

    // matrix data scanning (input)
//    printf("\nEnter row\t");     scanf("%d", &m1r);
//    printf("\nEnter column\t");  scanf("%d", &m1c);

/*    printf("\nEnter Data Values...\n");
    for(i=0;i<m1r;i++)
       for(j=0;j<m1c;j++)
          scanf("%5d", &mat1[i][j]);
*/
    printf("\nMatrix Looks like...\n");
    if((m1r==m2r) && (m1c==m2c))   //matrix addition conditon approved
here
    {

        for(i=0;i<m1r;i++)
        {
           for(j=0;j<m1c;j++)
               mat3[i][j]=mat1[i][j] + mat2[i][j];
        }
     }

    printf("\nMatrix Looks like...\n");
    for(i=0;i<m1r;i++)
    {
        for(j=0;j<m1c;j++)
          printf("%5d", mat3[i][j]);
        printf("\n");
    }
   return 0;
}
```

```
//matrix Subtraction (r1 = r2  and c1= c2)

#include <stdio.h>
int main()
{                              //matrix (2D array) declaration with
initialization
    int mat1[10][10]={{12,34,45,24},{56,67,78,39},{42,46,70,30}}, m1r=3,
m1c=4, i, j;
    int mat2[10][10]={{12,34,45,24},{56,67,78,39},{42,46,70,30}}, m2r=3,
m2c=4, mat3[10][10];

    // matrix data scanning (input)
//    printf("\nEnter row\t");     scanf("%d", &m1r);
//    printf("\nEnter column\t");  scanf("%d", &m1c);

/*    printf("\nEnter Data Values...\n");
    for(i=0;i<m1r;i++)
       for(j=0;j<m1c;j++)
          scanf("%5d", &mat1[i][j]);
*/
    printf("\nMatrix Looks like...\n");
    if((m1r==m2r) && (m1c==m2c))   //matrix subtraction conditon approved
here
    {

        for(i=0;i<m1r;i++)
        {
           for(j=0;j<m1c;j++)
               mat3[i][j]=mat1[i][j] - mat2[i][j];
        }
     }

    printf("\nMatrix Looks like...\n");
    for(i=0;i<m1r;i++)
    {
        for(j=0;j<m1c;j++)
          printf("%5d", mat3[i][j]);
        printf("\n");
    }
   return 0;
}
```

```
// Matrix Multiplication (r1 = r2  and c1= c2)

#include <stdio.h>
int main()
{                              //matrix (2D array) declaration with
initialization
    int mat1[10][10]={{1,3,4,4},{5,2, 4,3},{2,6,0,0}}, m1r=3, m1c=4, i,
j, k;
    int mat2[10][10]={{1,4}, {4,2},{5,6}, {7,3}}, m2r=4, m2c=2,
mat3[10][10];

    // matrix data scanning (input)
//    printf("\nEnter row\t");     scanf("%d", &m1r);
//    printf("\nEnter column\t");  scanf("%d", &m1c);

/*    printf("\nEnter Data Values...\n");
    for(i=0;i<m1r;i++)
       for(j=0;j<m1c;j++)
          scanf("%5d", &mat1[i][j]);
*/

//  formula   r1*c1   r2*c2      =>   if c1==r2 then resultant matrix
will looks like r1 *c2

    printf("\nMatrix Looks like...\n");
    if((m1c==m2r))   // Condition of matrix multiplication (approved
here)
    {
        for(i=0;i<m1r;i++)       // first mat row
        {
           for(j=0;j<m2c;j++)        //second mat column
           {
               mat3[i][j]=0;                                          //s =0;
               for(k=0; k<m2r; k++)
               {
                   mat3[i][j]= mat3[i][j] + mat1[i][k]*mat2[k][j];
               }
           }
        }
     }

    printf("\nMatrix Looks like...\n");
    for(i=0;i<m1r;i++)
    {
        for(j=0;j<m2c;j++)
          printf("%5d", mat3[i][j]);
        printf("\n");
    }
   return 0;
}
```

```
// String
// Define: String is a character array ended with a NULL character ('\0')

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#include<stdio.h>
int main()
{
 char a[10]={'a','y','a','n'};   // Q: is it a string or not ??
 int n=4;                         // A: chracter array but not a string

 for(i=0;i<n;i++)
   printf("%c", a[i]);
   return 0;
}
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#include<stdio.h>
int main()
{
 char a[10]={'a','y','a','n','\0'};   // Q: is it a string or not ??
 int n=4;                         // A: Yes this is a string

 for(i=0;i<n;i++)
   printf("%c", a[i]);
   return 0;
}
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#include<stdio.h>
int main()
{
 char a[10]="ayan";        // A: Yes this is a string
 i=0;
 while(a[i] != '\0')
 {
     printf("%c", a[i]);
     i++;
 }
   return 0;
}
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ String with pointer
#include<stdio.h>
int main()
{
 char a[10]="ayan", *p;
 p=a;           // array name means address of its first cell

 while(*p != '\0')
 {
     printf("%c", *p);
     p++;
 }
   return 0;
}
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ String with pointer
#include<stdio.h>
```

```
int main()
{
 char a[10]="ayan", *p;
 p=a;           // array name means address of its first cell
 printf("%s",*p);    // eliminate the loops
   return 0;
}
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Problem with %s
format
#include<stdio.h>
int main()
{
 char a[30];
 printf("Enter your name plz");
 scanf("%s", a);              // i/p:    Ayan Kumar Dey   // 14
characters     //a =  &a[0]
                              //  a will store   "Ayan"

 printf("Hi, %s, How are you",a);    // Expected o/p: Hi Ayan Kumar Dey,
How are you
                                     // Actual O/P:   Hi Ayan, How are
you
   return 0;
}
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Solution For %s
format
#include<stdio.h>
int main()
{
 char a[30];
 printf("Enter your name plz");
 scanf("%[^\n]s", a);              // %s = scan any characters untill
space or '\n'
                                 // %[^\n]s = scan any characters untill
'\n'
 printf("Hi, %s, How are you",a);    // Expected o/p: Hi Ayan Kumar Dey,
How are you
   return 0;
}
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Solution For %s
format (2nd approach)
#include<stdio.h>
int main()
{
 char a[30];
 printf("Enter your name plz");
 gets(a);
 puts("Hi,"); puts(a); puts(", How are you");
   return 0;
}
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Play with Pointers
and Strings
#include<stdio.h>
int main()
```

```
{
    char str1[]="ayan", str2[50];
    char *s= "Hello", *t;

    // array is very responsible to its values
    // pointer is irresponsible

    str2=str1;  // semantical error
    str1=str2; // semantical error
    s=t;           // correct statment
    t=s;
    s=str1;     //correct
    str1=s;   // error


    str2="mithun da";  // Error //array name =  address of a[0] = &a[0]
    t="Nachun Na";     // Correct
}
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Exercise of String
"Let Us C"
#include<stdio.h>
int main()
{
 printf(5+"Good Morning");   // 5 characters will be skipped from the
base address | Output: Morning
 printf("%c\n", "abcdefgh"[4]);      // 4th character considering 0 at
base index  | output 'e'
 return 0;
}

#include<stdio.h>
int main()
{
 char str1[]={'a','y','a','n','0'};
 char str2[]="ayan";
 printf("\n%s", str1);    //  output:   ayana
124443$%5h36y88%!@dt3472r31221u49326t46r27321
 printf("\n%s", str2);    //  output:   ayan
 return 0;
}

#include<stdio.h>
int main()
{
 char str1[]="ayan";
 char str2[]="ayan";

 if(str1==str2)      // values are same but address are diffrent
   puts("Bujhte perechis ?");
 else
   puts("kichui bujhis ni");             // output:  kichui bujhis ni
 return 0;
}
```

```c
#include<stdio.h>
int main()
{
 char str1[2]="A";
 printf("\n%c", str1[0]);  // output:  A
 printf("\n%c", str1[1]);  // output:  NULL
 printf("\n%s", str1);     // output:  A
 return 0;
}
```

//string

```
#include <stdio.h>
int main()
{
   printf(5+"Hello World"); output: World
   return 0;
}

#include <stdio.h>
int main()
{
   char str1[]="Hello", str2[]="Hello";
   if(str1==str2)
   printf("World");
   else
   printf("Energy");   // output: Energy
   return 0;
}

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

```
    printf("%c","ayan"[2]); // output:  a
    return 0;
}

#include <stdio.h>
int main()
{
    char str[8]="strings";
    char str1[7]="strings";
    printf("%c\n",str[0]);   // s
    printf("%s\n",str);      // strings
    printf("%s\n",str1);     // unpredicatble
    return 0;
}

#include <stdio.h>
int main()
{
    printf("%ld %ld %ld",sizeof(3),sizeof('3'),sizeof("3")); // dependent on compiler
32b compiler  2 2 2
    return 0;
}

// Difference between them ? is clear?
 char *p="ayan dey";
```

```
 char p[]="ayan dey";

// ASCII based codes




/*
#include <stdio.h>
#include <string.h>
int main()
{
   char str[20], str1[30], str2[]="ayan saha", str3[]="ayan das";
   int l;
   puts("\nenter a string");
   gets("%s",str);   // ayan dey

   l=strlen(str);   // 8
   strcpy(str1, str);   // copies the string from str to str1
   l=strcmp(str3, str2);   // abc - abd =>   97 98 99      abcd  -  abdx   l= (c-d)
                           //              97 98 100
```

```
                    //              0  0  -1
   // if l contains any non zero value then both string are not equal

   strrev(str1);
   strcat("ayan","dey");  => "ayandey"
   return 0;
} */



// WAP that will convert all small characters to capitals
// aYaN =>   AYAN
// a= 97 then z= 97+25 =122
// A=65 then Z= 65+25 = 90

#include <stdio.h>
int main()
{
   char str1[30], l, *p;
   puts("\nenter a string");
   gets("%s",str);
   p=str1;

   while(*p!='\0')
   {
```

```
        if(*p>=97 && *p<=122)   //small hand characters
          *p=*p-32;
        p++;
    }
}


//CAPS to small

#include <stdio.h>
int main()
{
    char str1[30], l, *p;
    puts("\nenter a string");
    gets("%s",str);
    p=str1;

    while(*p!='\0')
    {
        if(*p>=65 && *p<=90)   //caps characters
          *p=*p+32;
        p++;
    }
}
```

```
//TOGGLE case | AyaN  => aYAn

#include <stdio.h>
int main()
{
   char str1[30], l, *p;
   puts("\nenter a string");
   gets("%s",str);
   p=str1;

   while(*p!='\0')
   {
      if(*p>=65 && *p<=90)   // CAPS
        *p=*p+32;
      if(*p>=97 && *p<=122)    // scanning for small characters
        *p=*p-32;
      p++;
   }
}
```

```
// Play with Esc


#include <stdio.h>
int main()
{
  char ch;
  int a, b, c;
  do{
     puts("Program of Addition. Press Esc to exit from the loop..");
     scanf("%d%d", &a, &b);
     c=a+b;
     printf("\nsum is %d",c);


     puts("Do you want to continue? if yes press any key else Press escape to
exit..");
     scanf("%c", &ch);
  }while(ch!=27);
  return 0;
}
```

```
//palindrome  Checking || Input: Madam => Output: Yes
#include <stdio.h>
int strcmp1(char* s1,char* s2)
{
   int diff=0;
   while(*s1!='\0' && *s2!='\0')
   {
      if(*s1!=*s2)
        return *s1-*s2;
      s1++;
      s2++;
   }
   return diff;
}
int strlen1 (char *s)
{
   int l=0;
   while(*s++ != '\0')
   {
      l++;
   }
   printf("Length of the string is:%5d",l);
   return l;
}
char* strrev1(char* s)
{
   char *t;
   int i=0, len;
   len=strlen1(s);
   *(t+len)='\0';
   len--;
   while(*s != '\0')
   {
```

```
        *(t+len)=*s;
        len--;
        s++;
    }
    return t;
}
int main()
{
    char str[50], *str1, n;
    printf("\nEnter a string\n");
    scanf("%[^\n]s",str);

    puts("\nYou entered a string:\t"); puts(str);
    //printf("\nreverse of that string:\t%s",strrev1(str));

    if(!strcmp1(strrev1(str),str))
      puts("\n\nPALINDROME");
    else
      puts("\n\nNOT PALINDROME");
    return 0;
}
```

```
//signature
#include <stdio.h>
int count_space(char* s)
{
   int c=0;
   while(*s !='\0')
   {
      if(*s== ' ')
       c++;
      s++;
   }
   printf("\n%3d spaces are detected...\n",c);
   return c;
}
void mk_sign(char* s)
{
   int cs=count_space(s); //count no of spaces

   while(*s)
   {
      printf("%c. ",*s-32);
      while(*s!=32) //skip all chars untill a ' ' is found
        s++;
      cs--;    // reduce space counter
      if(cs==0)
      break;
      s++;  // skip the space
   }
   //printf("\n\n%s\n\n",*s);
   while(*s)
   {
      printf("%c",*s-32);
      s++;
```

```
    }
}
int main()
{
    char str[50]="ayan kumar dey";

    printf("\nEnter a string\n");
 //   scanf("%[^\n]s",str);


    puts("\nYou entered a string:\t"); puts(str);
    //printf("\nreverse of that string:\t%s",strrev1(str));

    mk_sign(str);
    return 0;
}
```

```
//word replacement
#include <stdio.h>
#include <string.h>
void replace (char* s, char* s1, char* s2)
{
   char temp[50];
   int i;
  while(*s)
  {
    //cut a word
    i=0;
    while(*s!=32)
    {
       temp[i++]=*s;
       s++;
    }
    temp[i]='\0'; //complete string
    //compare with s2 if found replace it else print it
    if(!strcmp(temp,s1))  // if good is found
      puts(s2);               // we'll print bad
    else
      puts(temp);
    s++;
  }
}
int main()
{
   char str[50]="Ayan is a good boy. Ratan is a good boy";
   char str1[50]="good", str2[50]="bad";
   replace(str, str1, str2);
   return 0;
}
```

```
#include <stdio.h>
void delay(int n)
{
   for(int i=0;i<n; i++)
      for(int j=0;j<n; j++)
         ;    // no work during this moment (                  // (15K * 15K)
}
int main()
{
   for(int i=0; i<5; i++)
   {
      printf("Hello World\n");
      delay(15000);
   }
   return 0;
}
```

```
//bitwise operator

// arithmatic op: + - * / %   2+3 => 5   4/2= 2

// logical op:   2 && 3 => 1   2 && 0 => 0

// Bitwise op:   works on bits   ( & bitwise AND, | bitwise OR,  ^ EXOR,  ~ NOT,  >> RIGHT SHIFT, << LEFT SHIFT)

//  a= 2   =>    0000 0000 0000 0010
//  b= 3   =>    0000 0000 0000 0011
// c= a & b; =>  0000 0000 0000 0010  (o/p will b 0 if any one of input is 0)
// output (c) = 2

//  a= 2   =>    0000 0000 0000 0010
//  b=3   =>     0000 0000 0000 0011
// c= a | b; =>  0000 0000 0000 0011  (o/p will b 1 if any one of input is 1)
// output (c) = 3

//  a= 2   =>    0000 0000 0000 0010
//  b=3   =>     0000 0000 0000 0011
// c= a ^ b; =>  0000 0000 0000 0001  (o/p will b 1 if both of the inputs are different)
// 0 0   0
// 0 1   1
// 1 0   1
// 1 1   0
// output (c) = 1

//  << left shift
//  >> right shift
// ~ one's complement   a= ~a;    1111 1111 1111 1101

#include <stdio.h>
int main()
{
   int a=5, b;
   printf("bitwise operator\n");
   b=a&5;                       //0000 0000 0000 0101
   printf("Value of b is %d\n",b); //0000 0000 0000 0101
   b=a|5;
   printf("Value of b is %d\n",b);
   b=~a;
   printf("Value of b is %d\n",b);
   return 0;
}


// 3 questions may arise
// NEED of bitwise operators =>  1. prog will be compact, 2. faster
// Ans: bit manipulations

// Q1: Set/reset bit (Conversion to 1 or 0)

   int a =39;  // 0000 0000 0010 0111
  // Q: check the 6th bit of your number. Is it 0 or 1 ??
```

```
// ans:
   int b=1; // 0000 0000 0000 0001
   b=b<<5;  // 0000 0000 0010 0000                    0110
   c=a&b;   // 0000 0000 0010 0000  non0 => 6th bit is SET //   0001    0000   zero  1st bit is 0
   c?printf("6th bit is 1"):printf("6th bit is 0");       //    0010    0010   non 0 2nd bit is 1
                                          //    0100    0100   non 0 3rd bit is 1
                                          //    1000    0000   zero  4th bit is 0
scanf(a);
scanf(pos);
b=1<<(pos-1);
a&b ? printf("pos bit is 1") : printf("pos bit is 0");


// Toggle bits
 int a= 5;  // 0000 0000 0000 0101
         // 1111 1111 1111 1010

// bitwise and operator is used for checking
// bitwise or operator is used to a SET a particular bit
// bitwise xor is used to toggle a particular bit

// Q2: Check bit
#include <stdio.h>
int main()
{
   unsigned int a=15;
   int i, mask, rslt;
   printf("bitwise operator\n");
   printf("In which index position, you want to check? (please enter within range 0 to 15)\t");
   scanf("%d", i);

   actual_pos= 1<<i;      //set index checker
   rslt=num&actual_pos;        //find
   rslt?printf(" 1"):printf(" 0");   //show
   return 0;
}



// ShowBits

#include <stdio.h>
int main()
{
   unsigned int a=15;
   int i, mask, rslt;
   printf("bitwise operator\n");
   for(i=15; i>=0; i--)
   {
      mask=1<<i;
      rslt=a&mask;
      rslt?printf(" 1"):printf(" 0");
   }
   return 0;
}
```

```
//switch case || alternative of nested if-else
#include <stdio.h>
int main()
{
   float v=3.5;    // v= 3.499999   or v= 3.500001  or v= 3.499998
   if(v==3.5)
     printf("Hello");
   else
     printf("Hi");
   return 0;
}
```

Output: "hi"

```
#include <stdio.h>
int main()
{
   float v=3.5f;    // v= 3.5
   if(v==3.5)
     printf("Hello");
   else
     printf("Hi");
   return 0;
}
```

```
#include <stdio.h>
int main()
{
   unsigned int a=32;
   int i, mask, rslt;
   printf("use bitwise operator\n");
   for(i=15; i>=0; i--)       // 16 bits(0-15)
   {
      mask=1<<i;
      rslt=a&mask;
      rslt?printf(" 1"):printf(" 0");   //using conditional operator
   }
   return 0;
}

// rslt?printf(" 1"):printf(" 0");   //using conditional operator
/*
//as same as
if(rslt)
  printf(" 1")
else
  printf(" 0");

// as same as
if(rslt==1)
  printf(" 1")
else
  printf(" 0");
*/



//i=15
// 32 means a=   0000 0000 0010 0000
//line 9: mask=  1000 0000 0000 0000     <=   0000 0000 0000 0001  << 15
//line 10: rslt= 0000 0000 0000 0000  => 0 => printf(" 0")

//i=14
// 32 means a=   0000 0000 0010 0000
//line 9: mask=  0100 0000 0000 0000     <=   0000 0000 0000 0001  << 14
//line 10: rslt= 0000 0000 0000 0000  => 0 => printf(" 0")

// ...

//i=5
// 32 means a=   0000 0000 0010 0000
//line 9: mask=  0000 0000 0010 0000     <=   0000 0000 0000 0001  << 5
//line 10: rslt= 0000 0000 0010 0000  => non zero => printf(" 1")

//....
```

```
//switch case|| alternative of nested if-else
#include<stdio.h>
int main()
{
flote v=3.5; //v= 3.499999 or v= 3.500001 or v= 3.499998
if(v==3.5)
printf("hello");
else
printf("hi");
return 0;
}
```

Output: "hi"

```
#include<stdio.h>
int main()
{
flote v=3.5f; //v= 3.5
if(v==3.5)
printf("hello");
else
printf("hi");
return 0;
}
```

Output: "hello"

```
/* FILE (text or binary)*/

// input: io.txt
// output:  out.txt
// a console: con (nick name)



// Case 1. user -> con  => con  (no files are used here) [will not b considered]
// Case 2. user -> con => out.txt
// Case 3. io.txt => out.txt    // file copy
// Case 4. io.txt => con

//Case 4.1:
#include <stdio.h>
int main()
{
   FILE *fp;    // FILE is a system defined data type. Only for files systems.
   char ch;
   fp=fopen("ram.txt","r");     //ram.txt   "hi ayanEOF"
   // "r" reading "w" writing "a" appending

   while(1)
   {
        ch=fgetc(fp);
        if(ch==EOF) break;
        printf("%c",ch);
   }
   fclose(fp);
   return 0;
```

```
}

//Case 4.2:  WAP that will count the no of characters, spaces, new lines, tabs, ',' an so on..
#include <stdio.h>
int main()
{
   FILE *fp;    // FILE is a system defined data type. Only for files systems.
   char ch;
   int nos=0, nota=0, nol=0, noc=0, nod=0;

   fp=fopen("ram.txt","r");     //ram.txt   "hi, .aya.na dey  EOF"
   // "r" reading "w" writing "a" appending

   while(1)
   {
        ch=fgetc(fp);
        if(ch==EOF) break;
        //printf("%c",ch);
        switch(ch)
        {
           case ' ': nos++; break;
           case '\t': notab++; break;
           case '\n': nol++; break;
           case ',': noc++; break;
           case '.': nod++; break;
           default:
        }
   }
   fclose(fp);
```

```
   printf("No of tabs are %d", nos);
   printf("No of tabs are %d", notab);
   printf("No of tabs are %d", nol);
   printf("No of tabs are %d", noc);
   printf("No of tabs are %d", nod);
   return 0;
}

// Case 3. io.txt => out.txt    // file copy
#include <stdio.h>
int main()
{
   FILE *fp;    // FILE is a system defined data type. Only for files systems.
   char ch;
   fp=fopen("io.txt","r");     //io.txt   "hi ayanEOF"
   if(fp==NULL)
   {
      printf("ERROR: SOURCE FILE NOT FOUND\n");
      exit(1);
   }

   ft=fopen("out.txt","w");
   if(ft==NULL)
   {
      printf("ERROR: TARGET FILE NOT FOUND\n");
      fclose(fp);
      exit(1);
   }
```

```
   while(1)
   {
        ch=fgetc(fp);
        if(ch==EOF) break;
        else
         fputc(ch, ft);
   }
   fclose(fp);
   fclose(ft);
   return 0;
}

// Case 2.1. user -> con => out.txt         // "Hi how are you" => out.txt
#include <stdio.h>
#include <string.h>
int main()
{
   FILE *fp;    // FILE is a system defined data type. Only for files systems.
   char s[100];

   fp=fopen("ram.txt","w");
   if(fp==NULL)
   {
      printf("ERROR: FILE NOT FOUND\n");
      exit(1);
   }

   printf("\nPlease Enter some texts\n");
```

```
   while(strlen(gets(s))>0)
   {
        fputs(s,fp);
        fputs("\n",fp);
   }
   fclose(fp);
   return 0;
}

// hi how are you
// r u from kolkata? yes or no ? ...
//



// Case 2.2. user -> con => out.txt          // out.txt => console
#include <stdio.h>
#include <string.h>
int main()
{
   FILE *fp;    // FILE is a system defined data type. Only for files systems.
   char s[100];

   fp=fopen("ram.txt","r");
   if(fp==NULL)
   {
      printf("ERROR: FILE NOT FOUND\n");
      exit(1);
   }
```

```
   while(fgets(s,99,fp))!=NULL)
     puts(s);

   fclose(fp);
   return 0;
}



// Binary file and recursion

//Break will exit the control from the loop or switch
//Return will exit the control from any function to Main
//Exit will return the control from any location to compiler
```

```c
// Prepprocessor

#include<stdio.h>
//#define me 25
int main()
{
   int i;
   //for(i=me; i<me*2; i++)
     // printf("%d",i);

    //printf("%ld",sizeof(i));

    if(sizeof(i)>-1)          //range of an int -32768 to +32767
      printf("hi");
    else
      printf("hello");
    return 0;
}

// Output:         hello
// Explaination:  4> unsigned integer  =>   4 < 0xFFFE    =>   "Hello"
```

```c
// Preprocessor

//.c => Preprocessor =>

#include<stdio.h>
#define me 25
int main()
{
    int i;
    for(i=me; i<me*2; i++)
        printf("%d",i);

     return 0;
}


int main()
{
    int i;
    for(i=25; i<25*2; i++)
        printf("%d",i);
     return 0;
}

```

```
/* FILE (text or binary)*/

// input: io.txt
// output:  out.txt
// a console: con (nick name)



// Case 1. user -> con  => con  (no files are used here) [will not b considered]
// Case 2. user -> con => out.txt
// Case 3. io.txt => out.txt    // file copy
// Case 4. io.txt => con

//Case 4.1:
#include <stdio.h>
int main()
{
   FILE *fp;    // FILE is a system defined data type. Only for files systems.
   char ch;
   fp=fopen("ram.txt","r");     //ram.txt   "hi ayanEOF"
   // "r" reading "w" writing "a" appending

   while(1)
   {
        ch=fgetc(fp);
        if(ch==EOF) break;
        printf("%c",ch);
   }
   fclose(fp);
   return 0;
```

```
}

//Case 4.2:  WAP that will count the no of characters, spaces, new lines, tabs, ',' an so on..
#include <stdio.h>
int main()
{
   FILE *fp;    // FILE is a system defined data type. Only for files systems.
   char ch;
   int nos=0, nota=0, nol=0, noc=0, nod=0;

   fp=fopen("ram.txt","r");     //ram.txt   "hi, .aya.na dey  EOF"
   // "r" reading "w" writing "a" appending

   while(1)
   {
        ch=fgetc(fp);
        if(ch==EOF) break;
        //printf("%c",ch);
        switch(ch)
        {
           case ' ': nos++; break;
           case '\t': notab++; break;
           case '\n': nol++; break;
           case ',': noc++; break;
           case '.': nod++; break;
           default:
        }
   }
   fclose(fp);
```

```
   printf("No of tabs are %d", nos);
   printf("No of tabs are %d", notab);
   printf("No of tabs are %d", nol);
   printf("No of tabs are %d", noc);
   printf("No of tabs are %d", nod);
   return 0;
}

// Case 3. io.txt => out.txt    // file copy
#include <stdio.h>
int main()
{
   FILE *fp;    // FILE is a system defined data type. Only for files systems.
   char ch;
   fp=fopen("io.txt","r");     //io.txt   "hi ayanEOF"
   if(fp==NULL)
   {
      printf("ERROR: SOURCE FILE NOT FOUND\n");
      exit(1);
   }

   ft=fopen("out.txt","w");
   if(ft==NULL)
   {
      printf("ERROR: TARGET FILE NOT FOUND\n");
      fclose(fp);
      exit(1);
   }
```

```
   while(1)
   {
        ch=fgetc(fp);
        if(ch==EOF) break;
        else
         fputc(ch, ft);
   }
   fclose(fp);
   fclose(ft);
   return 0;
}

// Case 2.1. user -> con => out.txt         // "Hi how are you" => out.txt
#include <stdio.h>
#include <string.h>
int main()
{
   FILE *fp;    // FILE is a system defined data type. Only for files systems.
   char s[100];

   fp=fopen("ram.txt","w");
   if(fp==NULL)
   {
      printf("ERROR: FILE NOT FOUND\n");
      exit(1);
   }

   printf("\nPlease Enter some texts\n");
```

```
   while(strlen(gets(s))>0)
   {
        fputs(s,fp);
        fputs("\n",fp);
   }
   fclose(fp);
   return 0;
}

// hi how are you
// r u from kolkata? yes or no ? ...
//



// Case 2.2. user -> con => out.txt          // out.txt => console
#include <stdio.h>
#include <string.h>
int main()
{
   FILE *fp;    // FILE is a system defined data type. Only for files systems.
   char s[100];

   fp=fopen("ram.txt","r");
   if(fp==NULL)
   {
      printf("ERROR: FILE NOT FOUND\n");
      exit(1);
   }
```

```
   while(fgets(s,99,fp))!=NULL)
      puts(s);

   fclose(fp);
   return 0;
}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//Record I/O in Files
// Dealing with Combo Pack of informations

// CODE: Storing information to FILE

#include <stdio.h>
// Making Combo pack of informations
struct emp{
   char name[40];
   int age;
   float bs;
};

int main()
{
   FILE *fp;
   char ch='y';
   struct emp e;       //define a sample of combo pack

   fp=fopen("emp.dat","w");   //dealing with binary files
   if(fp==NULL)
```

```
    {
        puts("\nERROR: CAN'T OPEN FILE\n");
        exit(1);
    }

    while(ch=='y')
    {
        puts("\nEnter your Name");  gets(e.name);
        printf("\nEnter your Age"); scanf("%d", &e.age);
        printf("\nEnter your Age"); scanf("%d", &e.bs);
        fprintf(fp,"%s,%3d,%5.2f",e.name, e.age, e.bs);

        puts("\nAdd another Record (Y/N)\t");          //requesting for another data
        fflush(stdin);
        ch=getche();
    }
    fclose(fp);
    return 0;
}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//Record I/O in Files
// Dealing with Combo Pack of informations

// CODE: Extracting information to FILE

#include <stdio.h>
// Making Combo pack of informations
struct emp{
```

```
    char name[40];
    int age;
    float bs;
};

int main()
{
    FILE *fp;
    char ch='y';
    struct emp e;       //define a sample of combo pack

    fp=fopen("emp.dat","w");   //dealing with binary files
    if(fp==NULL)
    {
        puts("\nERROR: CAN'T OPEN FILE\n");
        exit(1);
    }

    while(fread(&e, sizeof(e),1, fp)==1)
        printf("%s,%3d,%5.2f",e.name, e.age, e.bs);

    fclose(fp);
    return 0;
}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

// -reczise moves the pointer back by recsize bytes from the current positions
```

```
fseek(fp, -recsize, SEEK_CUR);

// Basically recsize or 0 are just the offsets that tell the compiler
// by how many bytes should the ppinter be moved from a particualr positions
// moves the pointer back by recsize bytes from the current positions

fseek(fp, 0, SEEK_END);
fseek(fp, recsize, SEEK_SET);



position=ftell(fp);

// ftell returns the position as a "long int" which is an offset from the begining of the file

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~`

//Recursion :
// Def: The process in which a function calls itself directly or indirectly is called
// recursion and the corresponding function is called as recursive function.
// E.g.: Sum of natural numbers
// f(n) = 1 + 2 + 3 +……..+ n
// Approach 1:
//    s=s+i;
// Approach 2:
//    f(n) = 1              n=1
//    f(n) = n + f(n-1)    n>1



// C code to implement Fibonacci series
```

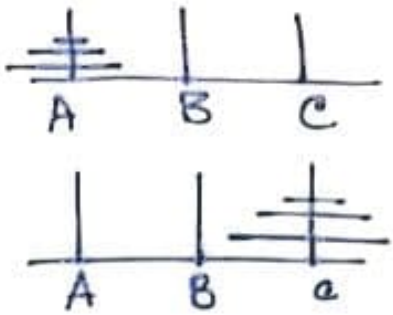
```
#include <stdio.h>

int fib(int n)  // Function for fibonacci
{
        if (n == 0)  // Stop condition
                return 0;

        if (n == 1 || n == 2)  // Stop condition
                return 1;

        else            // Recursion function
           return (fib(n - 1) + fib(n - 2));
}

int main()
{
        int n = 5;
        printf("Fibonacci series of %d numbers is: ",n);

        for (int i = 0; i < n; i++) {
                printf("%d ", fib(i));
        }
        return 0;
}
```

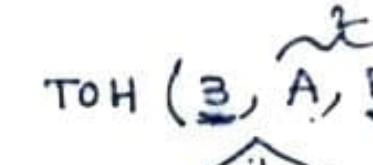
Prob: Towers of Hanoi

```
main()
{
  main();
}
```

Recursion → direct / Indirect

```
main()
{
  sum();
}
sum()
{
  main();
}
```





$f_{10} = f_9 + f_8$

$f_9 = f_8 + f_7$

$f_8 = f_7 + f_6$

:

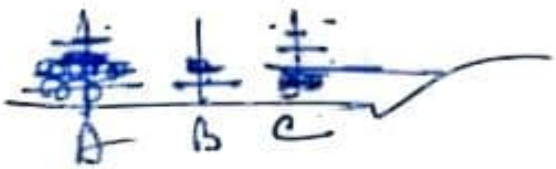
Fibb

0 1 1 2 3 5

$f_2 = f_0 + f_1$

$f_n = f_{n-2} + f_{n-1}$

$f_0 = 0, f_1 = 1$ $n \geq 2$.




```
TOH (n, A, B, C)
{
  if (n == 1)
    print (A → C);
  else
    TOH (n-1, A, C, B);
    print (A → C);
    TOH (n-1, B, A, C);
}
```

when to Stop

when to start